الفصل الاول

عنوان الموضوع:

النصوص البرمجية من جهة المخدم

الكلمات المفتاحية:

مخدم، نص برمجي، مخدم وب، زبون، تطبيق، تقنية، فهرس، استضافة، مستعرض.

ملخص:

لا بد من التمييز بصورة واضحة بين مفهوم النصوص البرمجية من جهة الزبون والنصوص البرمجية من جهة المخدم، وتحديد ما هي النقاط الإيجابية والسلبية الخاصة لكلٍ من المفهومين ومتى يصبح استخدام النصوص من جهة المخدم ضرورة.
ولكي نصل إلى إدراك أوسع لهذه المفاهيم، لابد من استيعاب بعض المفاهيم المرتبطة بها المرتبطة بها مثل مفهـــوم مخـــدمات الوب والاستضافة.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- تعريف النصوص البرمجية من جهة المخدم
- الاختلاف بين النصوص البرمجية من جهة المخدم والنصوص البرمجية من جهة الزبون
- مفهوم مخدمات الوب ومخدمات الوب الأكثرانتشاراً
- التقنيات واللغات المستخدمة في النصوص البرمجية من جهة المخدم
- كيفية إعداد مخدم IIS وإنشاء مجلدات افتراضية لاستضافة موقع

ما هو مخدم الوب؟

هو تطبيق مهمته استقبال طلبات مصدرها تطبيقات أخرى تدعى متصفحات الوب (أو زبون الوب)، وتقديم خدمة إرسال صفحات HTML مطلوبة من قبل هذه المتصفحات. يتم التواصل بين الزبون والمخدم اعتماداً على البروتوكول التطبيقي HTTP.

النصوص البرمجية من جهة المخدم

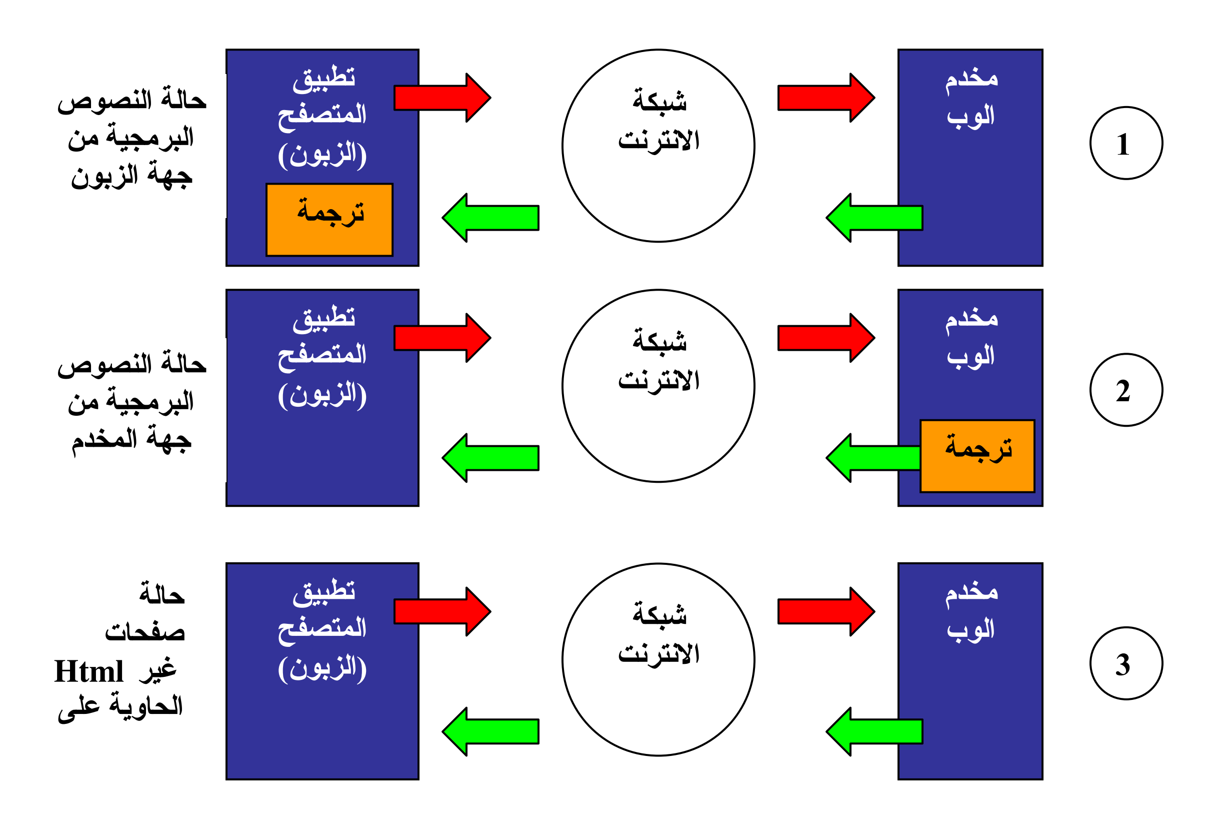
تعريف:

تُعتبر النصوص البرمجية من جهة المخدم إحدى تقنيات مخدمات صفحات الوب والتي يتم فيها الاستجابة ديناميكية.HTMLلطلب المستخدم والتفاعل معه، عبر تشغيل نص برمجي على المخدم وتوليد صفحات

تُستخدم النصوص البرمجية من جهة المخدم عادةً، لتأمين تخديم مواقع الانترنت التفاعلية التي تشكل واجهة تعامل مع قواعد بيانات أو أي مصدر آخر للبيانات.

تم تمييز هذه التقنية بعبارة (من جهة المخدم) بسبب وجود تقنيات أخرى تكون فيها مسؤولية تنفيذ النصوص البرمجية على عاتق متصفح صفحات الوب (التطبيق الزبون)، وتسمى التقنية باسم النصوص البرمجية من جهة الزبون.

جهة المخدم أم جهة الزبون ؟ (1)



لتوضيح فكرة النصوص البرمجية من جهة المخدم أو من جهة الزبون سنستعين بتمثيل بسيط لعلاقة مخدم الوب مع تطبيق زبون (المتصفح في حالتنا).

يمر التفاعل بين الزبون ومخدم الوب بالمراحل التالية:

- يرسل التصفح (التطبيق الزبون) إلى المخدم طلب HTTP عبر الشبكة، بهدف استعراض صفحة معينة باستخدام محدد الموارد القياسي (URL)
- يستلم المخدم الطلب ويبحث عن الملف المطلوب ليعيده عبر الشبكة إلى التطبيق الزبون

تمثل الأسهم المتحركة باتجاه المخدم في الشكل الموضح ضمن الشريحة، طلب الزبون والذي يحدد فيه الصفحة التي يريد استقدامها من مخدم الوب، في حين تمثل الأسهم المتحركة باتجاه التطبيق الزبون استجابة مخدم الوب وإرسال المحتوى المطلوب إلى الزبون.

جهة المخدم أم جهة الزبون ؟ (2)

نجد عند دراسة التفاعل بين مخدم الوب وزبون الوب، أننا أمام إحدى الحالات التالية :

الحالة 1− تحتوي الصفحة على نصوص برمجية من جهة الزبون؛

الحالة 2− تحتوي الصفحة على نصوص برمجية من جهة المخدم؛.

الحالة 3− لا تحتوي الصفحة على أية نصوص برمجية (تحتوي على عبارات HTML فقط).

ملاحظة:

يمكن أن تحتوي الصفحة على نصوص برمجية من جهة المخدم، وعلى نصوص برمجية من جهة الزبون، يجري عندها التفاعل بين زبون ومخدم الوب وفق الحالتين الأولى والثانية معاً.

نجد عند دراسة التفاعل بين مخدم الوب وزبون الوب، أننا أمام إحدى الحالات التالية :

الحالة 1 – تحتوي الصفحة على نصوص برمجية من جهة الزبون: تجري ترجمة وتنفيذ النصوص البرمجية المحتواة في الصفحة المطلوبة من قبل الزبون بعد استلامه للصفحة، نُطلق على هذه النصوص، النصوص البرمجية من جهة الزبون .

الحالة 2 – تحتوي الصفحة على نصوص برمجية من جهة المخدم: تجري ترجمة وتنفيذ النصوص البرمجية المحتواة في الصفحة التي أرسل الزبون في طلبها، من قبل المخدم قبل إرسال الصفحة، نُطلق على هذه النصوص، النصوص البرمجية من جهة المخدم .

الحالة 3– لا تحتوي الصفحة على أية نصوص برمجية (تحتوي على عبارات HTML فقط): يرسل المخدم الصفحة إلى الزبون الذي يستعرضها.

ملاحظة:

يمكن أن تحتوي الصفحة على نصوص برمجية من جهة المخدم، وعلى نصوص برمجية من جهة الزبون، يجري عندها التفاعل بين زبون ومخدم الوب وفق الحالتين الأولى والثانية معاً.

ما أهمية النصوص البرمجية من جهة المخدم؟ (1)

للإجابة على هذا السؤال ينبغي علينا الدخول في مقارنة بسيطة بين النصوص البرمجية من جهة المخدم والنصوص البرمجية من جهة الزبون من خلال سرد أهم النقاط الإيجابية والسلبية لكلٍ منهما.

النصوص البرمجية من جهة الزبون:

النقاط الإيجابية:

- تُساعد في دعم الحركة على الموقع باستخدام DHTML
- تتحمل جزء من حِمل المعالجة على المخدم حيث تتم عملية ترجمة وتنفيذ النص البرمجي على الحاسب الزبون
- تؤمن التفاعل مع المستخدم دون الحاجة إلى إعادة الاتصال مع المخدم (مثال عملية التأكد من بعض أخطاء البيانات المُدخَلة في نموذج معين)

النقاط السلبية:

- لا تملك القدرة على الوصول إلى أي مورد من موارد المخدم، ولا إلى أي تطبيق من التطبيقات المتصلة به مثل تطبيقات إدارة قواعد المعطيات. إذ يكون المصدر الوحيد للبيانات التي يمكن للنصوص البرمجية من جهة الزبون الوصول إليها، هي المعلومات المحتواة في الصفحة نفسها والتي تحوي، النص البرمجي، أو دخل المستخدم، أو معلومات من ملفات على جهاز الزبون (في حال تم منح الصلاحية للوصول إلى تلك المعلومات عن طريق المتصفح).
- توجد عدة لغات برمجة مُستَخدَمَة في كتابتها، مما يعزز فرص عدم توافقيتها الكاملة مع كافة المتصفحات ويجعل بعض المتصفحات غير قادرة على تصفحها
- لايتوفر الأمان الكافي عند استخدامها، إذ يمكن للزبون أن يستعرض بسهولة محتوى النصوص البرمجية من جهة الزبون لأنها تكون جزءاً من النص المصدري (الذي يمكن استعراضه بالخيار view source من خيارات المتصفح) مما يجعل عملية استخدام أي نوع من التحقق أوكلمات السر من جهة الزبون غير آمن بالشكل الكافي.

ما أهمية النصوص البرمجية من جهة المخدم؟ (2)

النصوص البرمجية من جهة المخدم:

النقاط الإيجابية:

– تستطيع –كونها تعمل من جهة المخدم– الوصول إلى موارد المخدم والتطبيقات المرتبطة به مثل تطبيقات إدارة قواعد المعطيات.

– يولد تطبيقها صفحات HTML قياسية يستطيع أي متصفح تفسيرها واستعراضها أياً كانت لغة البرمجة المستخدمة في كتابة هذه النصوص البرمجية.

– لا يستطيع الزبون استعراضها لأن ما يصل للمستخدم هو نص HTML ناتج عن تفسير وتشغيل النصوص البرمجية، مما يجعل محتوى النصوص البرمجية من جهة المخدم أكثر أماناً.

النقاط السلبية:

– تُعتبر عملية التفاعل في النصوص البرمجية من جهة المخدم بطيئة لأنها تتطلب الاتصال بالمخدم عند كل تفاعل.

- يتحمل المخدم عبء عمليات ترجمة هذه النصوص البرمجية.

التقنيات المستخدمة في النصوص البرمجية من جهة المخدم

من أشهر التقنيات المُستَخدَمة في تطوير النصوص البرمجية من جهة المخدم:

- PERL
- ASP
- ASP.NET
- PHP
- ColdFusion
- JSP

تاريخياً، جرى تطوير النصوص البرمجية من جهة المخدم، اعتماداً على لغات البرمجة التقليدية أو اللغات الخطاطية التقليدية مثل C و Perl و Shell script، بحيث كان التطوير يعتمد على إنشاء ما يسمى بـــ CGI (Common gateway interface) لتحصيل المعطيات التي يرسلها الزبون وللتعامل معها. وقد كان نظام التشغيل الذي يعمل عليه مخدم الوب، هو المسؤول عن عملية تنفيذ هذه البرامج بحيث كان يجري توجيه نتائج التنفيذ إلى تطبيق مخدم الوب.

جرى حديثاً تطوير تقنيات أخرى، كتلك التي تعتمد لغات مثل ASP و PHP، بحيث يجري تنفيذها مباشرةً من قبل مخدم الوب أو عن طريق وحدات برمجية إضافية وضمن فضاء العمل والعنونة المُخصصَين لمخدم الوب.

من أشهر التقنيات المُستَخدَمة في تطويرالنصوص البرمجية من جهة المخدم:

- PERL: لم يجر تصميم هذه التقنية لكتابة النصوص البرمجية الخاصة ببيئة الوب، ولكنها أثبتت فعالية عالية في هذا المجال، وهي ما تزال مستخدمة لكتابة بـــرامج CGI والوحـــدات الخاصـــة بمخـــدم الـــوب APACHE نظراً لإمكانياتها الكبيرة في معالجة سلاسل المحارف، وهي العناصر التي يتم تبادلهـــا بـــشكل أساسي في تطبيق وب. تأخذ ملفات النصوص البرمجية التي تستخدم هذه التقنية، اللاحقة PL.

- ASP: جرى تطوير هذه التقنية من قبل شركة مايكروسوفت، وهي تستخدم لغات مثل Java script و VB Script. تأخذ ملفات النصوص البرمجية التي تستخدم هذه التقنية، اللاحقة ASP.

- ASP.NET: جرى تطوير هذه التقنية أيضاً من قبل شركة مايكروسوفت، وركــزت علـــى اعتمـــاد البرمجة المقادة بالأحداث، واعتماد إطار عمل ".NET". تأخذ ملفات النصوص البرمجية التي تستخدم هـــذه التقنية، اللاحقة ASPX.

- PHP: جرى تطويرها كتقنية من تقنيات المصادر المفتوحة (Open source)، وهي تكافئ تقنيـــة ASP من حيث إمكانياتها، مع تمتعها بميزة الإنفتاح وبإمكانية التطوير والتحسين والتوسع من قبل العديد من الجهات نظراً لكونها مفتوحة المصدر. تأخذ ملفات النصوص البرمجية التي تستخدم هـــذه التقنيـــة، اللاحقـــة PHP.

- ColdFusion: وهي نسخة تجارية من التقنية التي طورتها شركة Macromedia لدعم النصوص البرمجية من جهة المخدم. تتوفر هذه التقنية على أكثر من بيئة عمل وتدعم التعامل مع أكثر من نظـــام إدارة قواعد بيانات. تأخذ ملفات النصوص البرمجية التي تستخدم هذه التقنية، اللاحقة CF.

- JSP: وهي تقنية مبنية على لغة جافا لبناء نص برمجي من جهة المخدم. تتوفر هذه التقنية على أكثر من بيئة عمل (Windows/Unix/Linux). تأخذ ملفات النصوص البرمجية التي تـــستخدم هـــذه التقنيـــة، اللاحقة JSP.

مخدمات الوب الأكثر انتشاراً

- مخدم Apache:



- مخدم IIS(Internet Information Services):



- مخدم Sun Java MicroSystem Web Server:



- مخدم Zeus:



خدم Apache:

يعتبر مخدم Apache، نظام مفتوح المصدر قدمته Apache software foundation ويتوفر مع رمازه مجاناً على منصات عمل Linux، Unix، Novell، Windows.

يتمتع هذا المخدم بالكثير من الخصائص المميزة نذكر منها: دعمه للعديد من لغات البرمجة مثل perl، php، ودعمه للبروتوكولات الآمنةSSL و TLS، وتوفيره لإمكانيات التحكم بشكل صفحات الخطأ، بالإضافة إلى توفر رمازه على نحو مفتوح مما يسمح بتطويره وتحسينه ومعالجة ثغراته بصورة أفضل وأسرع. يعد هذا المخدم من أكثر مخدمات الوب شعبيةً وانتشاراً على الإطلاق بحسب إحصاءات عام 2005.

مخدم IIS(Internet Information Services)

جرى تطوير هذا المخدم من قبل شركة Microsoft وهو عبارة عن مجموعة من الخدمات المخصصة لبيئة الوب والتي تعمل على نظام التشغيل Windows. يتم توزيع هذا التطبيق حالياً كخدمة من نظم التشغيل Windows 2000 و Windows 2003 server و Windows XP pro

تتضمن النسخة الحالية IIS 6.0 خدمات FTP, SMTP, NNTP, HTTP/HTTPS. ويُعتبر مخدم IIS المنافس الأقوى لمخدم Apache من حيث الشعبية والانتشار، ولكنه، وحتى نسخته الحالية، يعاني من بعض نقاط الضعف وخصوصاً من النواحي الأمنية. وقد جرى تجاوز العديد من الثغرات في النسخة التجريبية IIS 7.0 وجرى تصميمها على شكل وحدات ومكونات منفصل مما يضيف مرونة أكبر في التعامل مع هذه المكونات.

مخدم Sun Java MicroSystem Web Server :

جرى تطوير هذا المخدم من قبل شركة Sun MicroSystem يُدعى حالياً SUN ONE. يتميز هذا المخدم بخصائص أمان عالية، وبسهولة استخدام مما يجعله مخصصاً لتطبيقات العمل المتوسطة والكبيرة.

يتوفر المخدم على أغلب منصات العمل وهو يعطي العديد من الميزات للتطبيقات التي تستخدم تقنيات JAVA وJSP كما يدعم تقنيات ASP و PHP وتقنيات CGI.

مخدم Zeus:

تم تطوير هذا المخدم من قبل شركة Zeus technology، وهو يحتل المرتبة الأولى من حيث السرعة منذ عشر سنوات، ويعمل على منصات عمل UNIX بمختلف أنواعها.

تثبيت مخدم IIS

الخطوات التي تسبق عملية التثبيت:

- التأكد من تغطية عتاديات المخدم للحاجات الدنيا لنسخة IIS التي نثبتها. مثلاً، يُنصح باستخدام مخدم بذاكرة أولية 512 ميغابايت ومعالج P4 كحد أدنى، عند استخدام النسخة IIS 6.0.
- التأكد من تثبيت عائلة البروتوكولات TCP/IP وذلك عن طريق إعدادات خصائص الاتصال الشبكي.
- إعداد القرص الصلب الخاص بمخدم IIS وتهيئته باستخدام نظام الملفات NTFS.

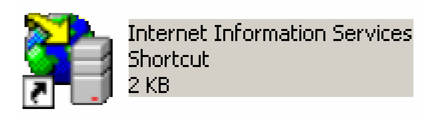
خطوات التثبيت:

تتضمن عدة إصدارات من نظام التشغيل Windows، المخدم IIS كإحدى خدماتها، لتثبيت هذا المخدم نتبع المراحل التالية:

1- نضغط زر Start ثم نختار خيار لوحة التحكم.
2- نختار خيار إضافة أو إزالة برامج ونحدد خيار إضافة أو إزالة مكونات Windows ونُفعّل خيار مخدم IIS.
3- يمكننا اختيار جزئيات التثبيت من خلال النقر على زر Details.

ملاحظة:

نلاحظ بالنقر على أيقونة Administrative tools من خيارات لوحة التحكم ظهور أيقونة خاصة للوصول إلى IIS.

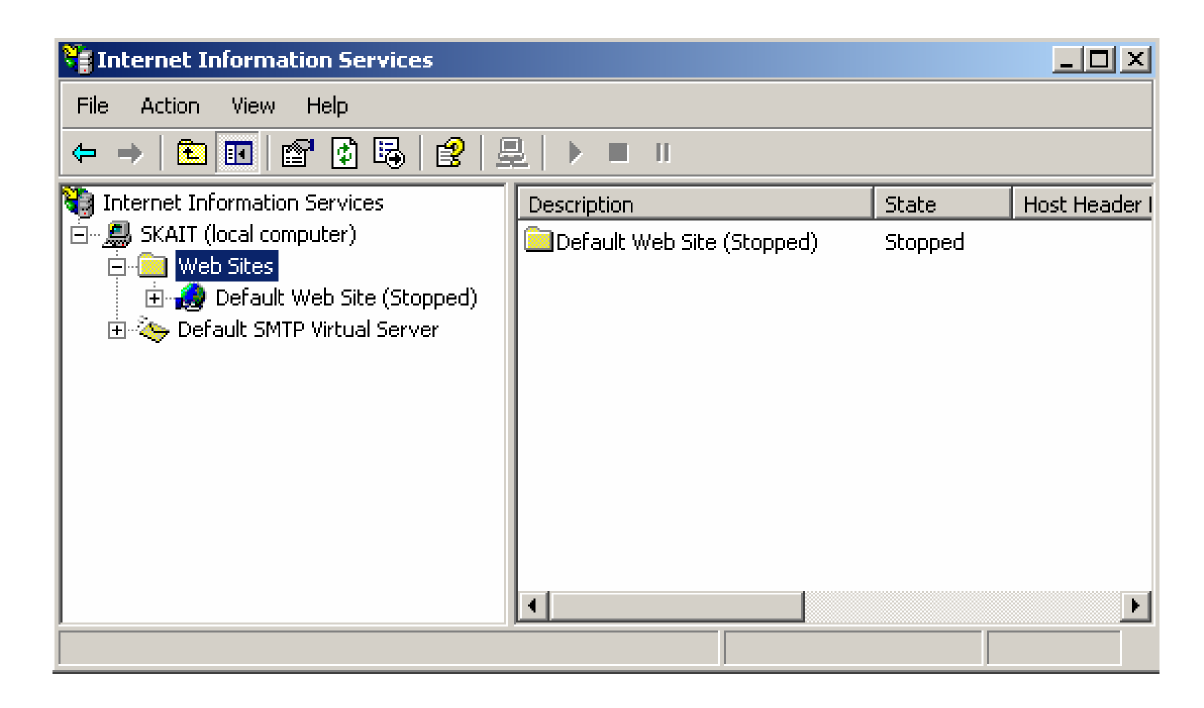


كما نلاحظ أيضاً أنه يجري تلقائياً عملية إضافة مجلد باسم INETPUB ضمن القرص الصلب الذي يجري تثبيت IIS عليه.

واجهة إدارة مجموعة مخدمات IIS

تتكون واجهة إدارة IIS، كما يظهر الشكل أدناه، من قسمين رئيسيين:

- القسم الأيسر الخاص بإظهار مخدمات IIS العاملة على نفس الجهاز أو على عدة أجهزة ضمن الشبكة، بالإضافة إلى الخدمات العاملة ضمن هذه المخدمات والمجلدات الرئيسية ضمنها.
- القسم الأيمن الذي تظهر فيه الفهارس الفرعية والملفات المحتواة ضمن هذه المجلدات.



يقدم IIS خدمات متعددة، فهو يعمل كمخدم SMTP وكمخدم FTP إضافة إلى عمله كمخدم وب. كما يمكننا تفعيل وإيقاف تفعيل أي من الخدمات باستخدام خيارات Start – Stop – Pause عبر النقر بالزر الأيمن على أية خدمة من هذه الخدمات.

ملاحظة:

يمكننا ضبط الإعدادات التلقائية لمخدم وب IIS بالنقر على الخيار Property من قائمة شريط الأدوات.

استضافة مواقع الوب في IIS

تعتمد الفكرة الأساسية في استضافة المواقع على إنشاء مجلدات تحتوي وثائق وصفحات الـ HTML المراد نشرها، بالإضافة إلى النصوص البرمجية التي تعمل من جهة الزبون أو من جهة المخدم. يتولى مخدم IIS دور الوسيط الذي يستقبل طلبات متصفح الوب ويعيد المحتوى المناسب من المجلد المناسب.

يُنشئ IIS مجلداً تلقائياً باسم wwwroot يعتبر المجلد الرئيسي التلقائي. كما يقدم IIS آلية لإضافة مجلـــدات أخرى وإدارتها. حيث تشكل هذه المجلدات جذر شجرة المجلدات التي تحتوي وثائق وصفحات مواقع الوب.

يكفي لنشر صفحة وب تسميتها وفق إسم الصفحة التلقائية المُحدد ضمن إعدادات مخدم الوب، ونقلهـــا إلـــى المجلد التلقائي المُحدد ضمن نفس إعدادات المخدم، بحيث يمكن لمتصفح الوب في الوصول إليها واستعراضها تلقائياً عد اتصاله بالمخدم.

ماهي المجلدات الافتراضية؟

لتمكين وصول مستخدمي المتصفح إلى صفحات موقع غير محتواة ضمن المجلد الرئيسي التلقائي يمكننا إنشاء ما يسمى بالمجلد الافتراضي.

يظهر المجلد للمتصفح وكأنه محتوى ضمن المجلد الرئيسي التلقائي بالرغم من كونه غير محتـــوى فيزيائيـــاً ضمنه.

يمتلك كل مجلد افتراضي اسم بديل (أو اسم مسار)، وهو الاسم الذي يستخدمه المستعرض للوصول إلى هـــذا المجلد ويكون عادةً أقصر من المسار الفيزيائي الحقيقي لهذا المجلد.

تؤمن المجلدات الافتراضية عدة مزايا، نذكر منها:

1- درجة أمان أعلى كونها تحجب عن المتطفلين، المسار الفيزيائي الحقيقي للمجلد.
2- سهولة عملية إدارة المخدم، وجعل عملية نقل الملفات فيزيائياً -من مكان إلى مكان آخر على القرص- عملية سهلة لا تسبب أي تغيير للعناوين المستخدمة ضمن الصفحات.
3- سهولة استذكار المسار الافتراضي كونه أقصر من المسار الفيزيائي.

إضافة مجلد افتراضي لاستضافة موقع

لإضافة مجلد افتراضي بغرض استضافة مجموعة من صفحات الوب، ننقر بالزر الأيمن على خيار Default Website ثم نختار من القائمة New الخيار Virtual directory. تؤدي هذه العملية إلى تـــشغيل معـــالج خاص بإنشاء المجلدات الافتراضية.

- يُطلب تحديد الاسم المُستَخدَم للوصول إلى هذا المجلد؛

- يُطلب تحديد اسم الموقع الفيزيائي لهذا المجلد على القرص؛
- يُطلب تحديد صلاحيات الوصول لمحتويات هذا المجلد.

بعد ضبط هذه الإعدادات يصبح بإمكاننا وضع ملف من نمط HTML ضمن المجلد الفيزيائي الحقيقي الـــذي يمثله المجلد الافتراضي ويصبح بإمكاننا استعراضه عن طريق متصفح الوب، حيث يمكننـــا الوصـــول إلـــى محتوى صفحة الوب مباشرة باستخدام محدد الموارد القياسي URL.

مثال:

يجري الوصول إلى صفحة test.html على مجلد افتراضي باسم myweb ضمن مخدم موصول على الشبكة ويحمل العنوان IP: 10.12.17.5، بكتابة محدد المورد القياسي:

http://10.12.17.5/myweb/test.html

فإذا كان لدينا مخدم DNS، نستطيع ربط http://10.12.17.5/myweb باسم نطاق معين فيـــصبح محـــدد الموراد القياسي من الشكل:

http://mydomain.com/test.html

الفصل الثاني

عنوان الموضوع:

النماذج في XHTML

الكلمات المفتاحية:

نموذج، واصفة، تأشيرة، عنصر، قيمة، حدث، إرسال، حقل نصي، حقل اختيار، زر، إعادة تأهيل.

ملخص:

النماذج في XHTML

تُعدّ النماذج واجهة التفاعل الأساسية بين المستخدم والتطبيقات العاملة على المخدم والنصوص البرمجية من جهة المخدم. تغطي هذه الجلسة النماذج وعناصرها المختلفة.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- لمحة عن XHTML
- النماذج في XHTML
- عناصر نماذج XHTML
  - الحقول النصية
  - حقول الاختيار
  - الأزرار
  - الحقول المخفية

ما هي XHTML ؟

هو عبارة عن معيار متقدم عن المعيار HTML4.0 جرى وضعه لبناء نسخة HTML متوافقة تماماً مع الشروط الصارمة للمعيار XML من ناحية أسلوب كتابة التأشيرات، واحترام إغلاق كل تأشيرة مفتوحة، والتركيز على التداخل الصحيح للتأشيرات، وعدم التسامح في تجاوز بعض التأشيرات، واستخدام الفواصل: (" ") لتحديد قيم الواصفات مثل "Attribute="Value، وغيرها.

النماذج في XHTML

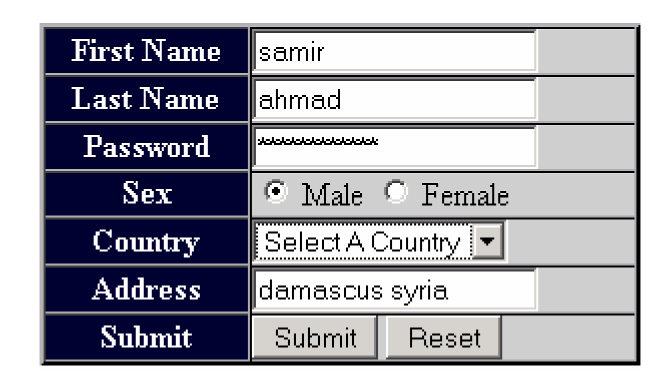
الغرض من النماذج:

- تُعرَّف النماذج بأنها آلية تهدف إلى جعل صفحات HTML أكثر تفاعلية. وتسمح النماذج لمستخدمي صفحة الوب بتوزيع البيانات ضمن حقول محتواة في صفحة HTML وإرسال هذه البيانات إلى مخدم الوب حيث تجري معالجتها
- يتدرج تعقيد تفاعلية النماذج من حقل بحثٍ بسيط، إلى نموذج خاص بنظام بيع وشراء متكامل عبر الشبكة، أو إلى نظام استعلام إلكتروني، أو أية خدمة تمكِّن المستخدم من إدخال معلومات تمهيداً لمعالجتها

مكونات النموذج:

- يتكون النموذج من حقل إدخال أو أكثر. يمكن أن تكون هذه الحقول، حقول إدخال نصية، أو أزرار، أو مربعات اختيار ، أو قوائم، أو حتى خرائط صور
- تكون عناصر النموذج محصورةً بين التأشيرتين: <form> </form>
- يحتوي النموذج مكونات HTML أخرى، فهو لايقتصر بالضرورة على عناصر النموذج. فعلى سبيل المثال، يمكن للنموذج أن يحتوي نصوصاً وصوراً (تساعد في شرح كيفية ملء حقول النموذج). كما يمكن للنموذج أن يحتوي على نصوص برمجية من جهة الزبون، مكتوبة بلغة JavaScript أو غيرها، تساعد في عملية تقييم البيانات

عند إرسال النموذج للمعالجة، يُرسل المتصفح بيانات الحقول إلى المخدم لمعالجتها أو إلى عنوان بريد إلكتروني أو إلى نص برمجي من جهة الزبون. يُعبر الشكل التالي عن نموذج تقليدي:



عناصر النموذج: الواصفة Method

تحدد هذه الواصفة أسلوب إرسال البيانات وطريقة إرسال طلب HTTP. تُميِّز هذه الواصفة حالتين:

- **حالة نموذج يستخدم طريقة POST**: حيث يجري توجيه المتصفح لإرسال البيانات إلى مخدم وب أو إلى عنوان بريد إلكتروني، ضمن أغراض خاصة يتضمنها الطلب HTTP المُرسَل إلى المخدم.
- **حالة نموذج يستخدم الطريقة GET**: حيث يجري توجيه المتصفح لإرسال البيانات إلى مخدم وب أو إلى عنوان بريد إلكتروني، على شكل سلسلة محارف تضاف إلى المُحدد URL ضمن طلب HTTP المُرسَل إلى المخدم.

بالإضافة إلى طريقتي GET و POST هناك العديد من الطرق الأخرى لإرسال طلب HTTP مثل الطرق PUT، Delete، Trace، Connect، لكننا سنتعامل حصرياً مع الطريقتين GET و POST، وسنستعرض لاحقاً الفروق الأساسية بين هاتين الطريقتين.

عناصر النموذج: الواصفة Action

- تحدد الواصفة Action للمستعرض العنوان الذي يجب أن تصل إليه البيانات المُرسَلة، إذ يمكن أن تكون القيمة المُسنَدة للواصفة، عنوان بريد إلكتروني، أو محدد URL لصفحة تحتوي على نص برمجي من جهة المخدم.
- تتم عملية الإرسال، والوصول إلى العنوان المحدد في الواصفة Action، عند نقر زر الإرسال

.Submit

مثال 1:

```
<FORM METHOD="POST" ACTION="SK@scs-net.org">
… Content
</FORM>
```

يقوم هذا النموذج، عند ضغط زر Submit، بحزم البيانات وإرسالها كرسالة بريد إلى عنوان البريد المحدد في المثال SK@scs-net.org.

مثال 2:

```
<FORM METHOD="POST" ACTION="getInfo.aspx">
… Content
</FORM>
```

يقوم هذا النموذج، عند ضغط زر Submit، بحزم البيانات وإرسالها إلى الملف getInfo.aspx على المخدم والذي يحتوي النص البرمجي الخاص بمعالجة البيانات المُرسَلة.

مثال 3:

```
<FORM METHOD="GET" ACTION="Test.php">
… Content
</FORM>
```

يقوم هذا النموذج، عند ضغط زر Submit، بإرسال إلى ملف Test.php وذلك بإضافة البيانات إلى محدد URL الخاص بالصفحة، فإذا كانت إحدى الحقول المراد إدخالها هي name وكانت قيمة الحقل هي sami، يظهر محدد URL عند الإرسال، بالشكل التالي:

```
http://myDomain.com/Test.php? name=sami
```

عناصر النموذج: الواصفة EncType

- تحدد الواصفة EncType نمط الترميز المُستَخدَم عند إرسال بيانات النموذج. تأخذ هذه الواصفة قيمها من أنماط المعيار MIME، وهو معيار يوصِّف أسلوب ترميز البيانات، جرى تطويره لترميز البيانات المختلفة ضمن شكل نصي وباستخدام حروف الأبجدية.

- يجري تحديد نمط الترميز على النحو التالي: MIMEType/MIMESubType. فعلى سبيل المثال يظهر نمط الترميز التلقائي والذي يُعبِّر عن القيمة التلقائية للواصفة EncType كما يلي:

```
<Form ENCTYPE=application/x-www-form-urlencoded>
```

حيث يصلح الترميز السابق لجميع نماذج الوب عدا تلك التي تتطلب إرسال ملفات إلى مخدم الوب، والتي يجري فيها استخدام القيمة:

```
<Form ENCTYPE=multipart/form-data>
```

عناصر النموذج: الواصفة AcceptCharSet

- تحدد الواصفة AcceptCharSet قائمة المحارف المتاح استخدامها في النموذج، وهي واصفة ضرورية خصوصاً في النماذج متعددة اللغات. تكون القيمة التلقائية لهذه الواصفة UNKNOWN، التي تشير إلى استخدام نفس قائمة المحارف المستخدمة في ترميز النموذج.

- عند الحاجة لاستخدام أكثر من قائمة محارف، يمكن وضع قيم الواصفة السابقة على شكل قيم متتالية يجري فصلها بفاصلة، كما هو الحال في المثال التالي:

```
<Form AcceptCharset="windows-1256,iso-8859-1">
```

عناصر النموذج: الواصفة Target

تُستخدم الواصفة Target في بيئة الصفحات متعددة الأطر. تُعبر قيمة هذه الواصفة عن اسم الإطار الهدف الذي ستظهر فيه الإجابة بعد إرسال النموذج (في حال أعاد النص البرمجي من جهة المخدم أي خرج). فعلى سبيل المثال، تُستخدم هذه الواصفة عند الحاجة لإدخال بيانات خاصة بنموذج بحث متوضع ضمن إطار ما في صفحة HTML، وعند الحاجة لاسترجاع نتيجة البحث ضمن إطار آخر.

تأخذ الواصفة target القيم:

- **"_blank"**: تجري إعادة خرج النموذج في نافذة جديدة بدون اسم.
- **"_self"**: تجري إعادة خرج النموذج ضمن نفس الإطار الذي يحوي النموذج.
- **"_parent"**: تجري إعادة خرج النموذج ضمن الإطار الأب للإطار الذي يحوي النموذج.
- **"_top"**: تجري إعادة خرج النموذج ضمن الإطار الرئيسي مع إزالة كل الإطارات الأخرى.

مثال:

```
<form action="SomeThing.aspx" method="POST" Target="_blank" >
```

الأحداث ONSUBMIT و ONRESET:

- يجري تنفيذ الحدث **ONSUBMIT** عند إرسال النموذج بضغط زر submit.
- يجري تنفيذ الحدث **ONRESET** عند إلغاء معلومات النموذج بضغط reset .

مثال:

```
<form action=”test.aspx”  method=”POST” onsubmit= ”window.alert(‘submitted successfully’);”>
```

تظهر، عند إرسال هذا النموذج، الرسالة ”submitted successfully“

عناصر النموذج: واصفات أخرى

لا تقتصر واصفات وأحداث النماذج على تلك التي قمنا بشرحها في الشرائح السابقة، إذ توجد واصفات وأحداث أخرى سنذكر بعضها فيما يلي:

من هذه الواصفات:

- **ID**: تُستَخدَم هذه الواصفة لتحديد اسم فريد للنموذج.

- **Class**: تُستَخدَم هذه الواصفة في حال الرغبة باستعمال الأنماط الخاصة بملفات CSS المُعرَّفة ضمن التأشيرة Style من نفس الوثيقة.
- **Style**: تُستَخدَم هذه التأشيرة لتعريف أنماط يجري استخدامها في تنسيق عناصر الوثيقة.
- **Title**: تُستَخدَم لتحديد عنوان العنصر حيث يستفيد المتصفح من قيمة هذه الواصفة لإظهار مربع يحتوي هذه القيمة لدى مرور مؤشر الفأرة فوق العنصر.

ومن هذه الأحداث:

- **ONCLICK** يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند النقر بالزر الأيسر فوق العنصر
- **ONDBLCLICK** يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند النقر المزدوج بالزر الأيسر فوق العنصر
- **ONMOUSEDOWN** يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند الضغط بالزر الأيسر فوق العنصر
- **ONMOUSEUP** يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند تحرير الضغط عن الزر الأيسر فوق العنصر
- **ONMOUSEOVER** يتم يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند وضع مؤشر الفأرة فوق العنصر

- **ONMOUSEMOVE** يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند تحريك مؤشر الفأرة فوق العنصر
- **ONMOUSEOUT** يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند خروج مؤشر الفأرة من المساحة التي يحددها العنصر
- **ONKEYPRESS** يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند ضغط وتحرير مفتاح فوق العنصر
- **ONKEYDOWN** يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند ضغط مفتاح فوق العنصر
- **ONKEYUP** يجري تنفيذ الإجرائية المُسنَدة لهذا الحدث، عند تحرير مفتاح فوق العنصر

عناصر النموذج

مثال:

```
<form method=”POST” action=”Test.aspx”
target=”_self”  ID=”myForm”
class=”normalForm”  EncType=”multipart/form-data”  dir=”ltr”
onclick=”window.alert(‘you clicked the form’)”>
… Contents
</form>
```

- يستعرض هذا المثال نص HTML يُعرِّف نموذج يعتمد على الطريقة POST في إرسال البيانات
- يحدد النموذج الملف test.aspx، كملف هدف يحتوي نصاً برمجياً من جهة المُخدم ويجري إرسال البيانات إليه للمعالجة
- تظهر الاستجابة على هذا النموذج (كنتيجة لتنفيذ الملف test.aspx) ضمن نفس الإطار الذي يظهر فيه النموذج، لأن النموذج استخدم القيمة ”_Self“
- يمتلك النموذج السابق الإسم ”myForm“ ويجري تطبيق النمط ”normalForm“ على هذا النموذج (لا بد أن يكون هذا النمط معرف مسبقاً ضمن وثيقة CSS أو ضمن نفس الوثيقة باستخدام التأشيرة STYLE)
- يُستخدَم في هذا النموذج الترميز ”multipart/form-data“ مما يعني أن النموذج يُرسل محتوى ملف إلى المخدم
- تظهر الرسالة ”you clicked the form“ عند النقر على النموذج

حقول النماذج: الحقول النصية

يمكن استخدام عدة أنواع من الحقول النصية في النموذج نستعرضها فيما يلي:

حقل إدخال نص:

- تُعتبر حقول إدخال النصوص من أكثر أنواع الحقول استخداماً في النماذج. يتألف حقل إدخال النص من حقل فارغ مخصص لإدخال سطر وحيد
- يجري تحديد حجم هذا الحقل باستخدام الواصفة Size، ويجري تحديد عدد المحارف المسموح إدخالها بالواصفة maxlength حيث تقاس Size و maxlength بعدد المحارف
- يمكن أن تكون الواصفة maxlength أكبر من الواصفة Size. في هذه الحالة يجري زلق محتويات الحقل النصي أثناء الكتابة
- يمكن استخدام الواصفة Value، لإعطاء قيمة تلقائية للحقل

لا يوجد في HTML أية آلية مباشرة للتحكم بنوع المدخلات، لذا نلجأ إلى النصوص البرمجية من جهة الزبون، أو إلى النصوص البرمجية من جهة المخدم، لتقييم البيانات وإظهار رسائل الخطأ في حال عدم تطابقها مع نمط أوتنسيق البيانات المطلوبة.

مثال:

لإنشاء حقل نصي باسم mySample يحتوي على قيمة تلقائية هي test، ويكون بطول 10 محارف، وبعدد أقصى من المحارف يبلغ 40 محرفاً، نكتب النص التالي:

```
<input    type="text"    name="mySample"    value="test"    size="10"
maxlength=40 />
```

حقول النماذج: الحقول النصية

حقل إدخال كلمة السر:

- يُستَخدَم هذا النمط من الحقول النصية كالحقول النصية العادية مع تطبيق قناع يخفي المحارف المُدخَلة.

إذ لا يتمكن الناظر إلى الشاشة من رؤية المحارف، بل يرى عوضاً عنها محرف القناع الذي غالباً ما يكون أحد المحارف "O" أو "*"

- يجري إظهار المحرف القناع ("O" أو "*") أثناء عملية الإدخال أمام الناظر إلى الشاشة، ولكن هذا لا يعني أن المحارف تكون مُقنعة بالنسبة للبرامج العاملة على الجهاز الذي تجري عليه عملية الإدخال أو أثناء عملية إرسال البيانات إلى المخدم.

مثال:

```
<input type="password" name="myPass" size="10" maxlength=25 />
```

حقول النماذج: الحقول النصية

حقل إدخال اسم ملف:

- يسمح هذا النمط من حقول الإدخال النصية للمستخدم باختيار ملف حتى يجري إرساله إلى المخدم
- يكون اسم الملف محتوى في نص يعبر عن المسار الذي استخدمه المتصفح للوصول إلى الملف
- يقوم هذا النمط بإظهار الحقل النصي، إضافةً إلى زر يساعد في استعراض المجلدات بحثاً عن الملف المطلوب
- ليعمل هذا الحقل، لا بد من استخدام الطريقة POST في إرسال بيانات النموذج وعدم استخدام الطريقة GET
- يمكن استخدام الواصفة Accept مع هذا النمط للسماح بإرسال نمط معين من الملفات. تحدد قيم الواصفة Accept بنمط MIME للملفات المراد إرسالها

مثال:

```
<input type="file" size=8 name="myFile" accept="text/*"  />
```

في هذا المثال، يجري إنشاء حقل من نمط حقل ملف بإسم myFile، بحيث يكون عرض الحقل 8 محارف، وبحيث يقبل هذا الحقل الملفات من النمط Text بجميع أنماطه الفرعية مثل (text/html, text/css text/plain).

عناصر النماذج: الحقول النصية

حقل إدخال نص متعدد الأسطر :

يساعد هذا النمط من الحقول في إدخال نص متعدد الأسطر وذلك باستخدام التأشيرة <TEXTAREA>.

من أهم الواصفات الخاصة بهذه التأشيرة:

- الواصفة Rows التي تحدد عدد الأسطر
- الواصفة Cols التي تحدد عدد الأعمدة
- الواصفة wrap التي تحدد قابلية النص للالتفاف عند تجاوز عرض مساحة النص. يمكن لقيمة الواصفة أن تأخذ إحدى القيم "off"، أو "virtual"، أو "Physical"، حيث تساعد القيمة "off" في تعطيل التفاف النص، بينما تساعد القيمة "virtual" في تثبيت التفاف النص ولكنها لا تقوم بإضافة أي فاصل إلى البيانات التي يجري إرسالها بعكس القيمة "Physical" التي تضيف فاصل سطر حقيقي عند كل التفاف للنص ويجري إرساله مع البيانات عند إرسال النموذج.

مثال:

```
<TEXTAREA rows="10" cols="6" name="myTextArea" wrap="off">
… This is a test
</TEXTAREA>
```

يُنشئ هذا المثال مساحة نصية بعرض 6 محارف وارتفاع 10 محارف باسم myTextArea ويجري في هذا النموذج تعطيل خاصة التفاف النص.

حقول النماذج: حقول الاختيار

يندرج تحت هذا التصنيف الحقول ذات الأنماط التالية:

مربعات التحقق:

- يُنشئ هذا النمط من الحقول عنصر يمكن تفعيله أو إزالة تفعيله
- يكون الشكل التقليدي لهذا النمط عبارة عن مربع تظهر فيه إشارة X عند اختياره
- عند إرسال النموذج، يجري إرسال قيم العناصر المُختارة وهي قيم العناصر التي تظهر أمامها إشارة X. لذا يجب أخذ هذا الموضوع في الحسبان أثناء كتابة النص البرمجي الذي يعالج البيانات المُرسَلة
- تعتمد القيمة التي يجري إرسالها في هذا النمط من الحقول، على القيمة المُسندة للواصفة Value
- يمكن تفعيل حقل التحقق تلقائياً عند إظهار النموذج وذلك عبر إضافة الواصفة "Checked" إلى التأشيرة. لا تأخذ الواصفة "Checked" أية قيمة ولكن، للالتزام بقواعد كتابة الواصفات، نُسند لهذا النوع من الواصفات قيمة مساوية لإسم الواصفة ونكتبها: "Checked="Checked

مثال:

```
<input type="checkbox" value="test" name="myCheck"
Checked="Checked" />
```

حقول النماذج: حقول الاختيار

أزرار الراديو:

- يشبه شكل هذا النمط من الحقول، شكل خيارات قنوات الراديو حيث يمكن اختيار خيار واحد فقط من الحقول التي تنتمي إلى مجموعة واحدة
- تعتمد القيمة التي يعيدها النموذج، كما هي الحال مع مربعات التحقق، على القيمة المُسندة للواصفة Value
- يمكن تفعيل حقل التحقق تلقائياً عند إظهار النموذج وذلك عبر إضافة الواصفة "Checked" وإعطائها القيمة "Checked" كما هو الحال في حالة مربعات التحقق

مثال:

```
<input type="radio" name="myRadio" vaue="option1" checked="checked"
/> first option<br>
<input type="radio" name="myRadio" vaue="option2" /> second option
<br>
<input type="radio" name="myRadio" vaue="option3" /> third option
```

يظهر عند تنفيذ النص أعلاه النتيجة التالية، حيث يجري إرسال القيمة "first option"، وفقاً للاختيار المُبين في الشكل:

first option
second option
third option

عناصر النماذج: حقول الاختيار

قوائم الاختيار:

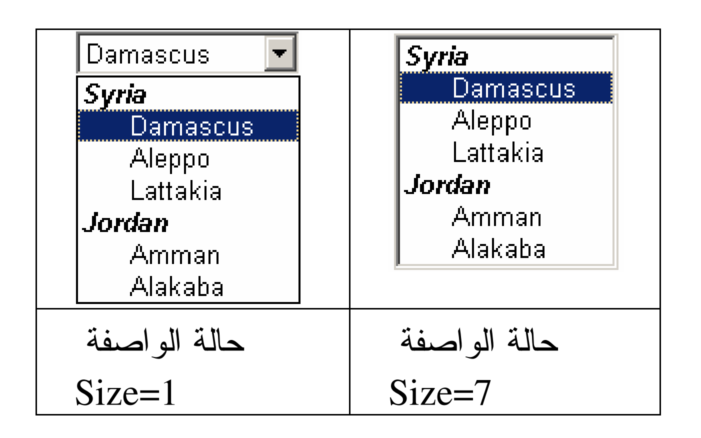
- تستخدم قوائم الاختيار التأشيرة <Select> لإنشاء نمطين أساسيين من القوائم:
  - القوائم المنسدلة

○ القوائم القابلة للزلق

- يساعد كلا النوعين في تنفيذ عملية اختيار خيار واحد أو عدة خيارات من قائمة خيارات
- يجري تحديد عناصر القائمة بالتأشيرة <option> حيث يجري تضمين كل خيار ضمن القائمة باستخدام التأشيرة <option></option>
- يجري إرسال المتحول المُحدد يالواصفة name، عند إرسال النموذج، محملاً بقيمة الخيار المحدد بـــ option في حال قام المستخدم باختيار وحيد. أما في حالة الاختيار المتعدد، فيجري تحميل متحول يمثل قائمة خيارات مفصولة عن بعضها البعض بفاصلة
- يكفي إضافة الواصفة multiple وإعطائها القيمة "multiple" لتحديد إمكانية استخدام عدة خيارات في قائمة الاختيار المعنية
- يستطيع المستخدم، عند تفعيل خاصة الاختيار المتعدد، اختيار أكثر من عنصر في القائمة بضغط زر CTRL والنقر على الخيارات المطلوبة. أما في حال عدم تفعيل هذه الخاصة، فإن قوائم الاختيار تعمل بنفس الآلية التي تعمل بها أزرار الراديو حيث تسمح باختيار خيار واحد فقط من مجموعة خيارات
- تتحكم قيمة الواصفة Size بارتفاع قائمة الاختيار القابلة للإنزلاق، وعندما تُسند إلى هذه الواصفة القيمة "1"، يجري إظهار سطر واحد من القائمة، وتتحول هذه القائمة إلى قائمة منسدلة
- لا توجد واصفة خاصة للتحكم بعرض قوائم الاختيار إنما تأخذ القائمة العرض اللازم لاحتواء الخيار صاحب العدد الأكبر من من المحارف
- يجري تحديد خيار تلقائي أومجموعة من الخيارات التلقائية باستخدام الواصفة Selected وبإعطاءها القيمة "Selected"، وذلك ضمن التأشيرة option لكل خيار من الخيارات المعنية
- يمكن إنشاء مجموعات فرعية من الخيارات ضمن قائمة الخيارات الواحدة وذلك باستخدام التأشيرة <Optgroup> وحصرجميع خيارت المجموعة الفرعية ضمن هذه التأشيرة

مثال:

```
<select name="select">
<optgroup label="Syria">
<option value="1">Damascus</option>
<option value="2">Aleppo</option>
<option value="3">Lattakia</option>
</optgroup>
<optgroup label="Jordan">
<option value="4">Amman</option>
<option value="5">Alakaba</option>
</optgroup>
</select>
```



حالة الواصفة Size=1	حالة الواصفة Size=7

عناصر النماذج: الأزرار

هناك ثلاثة أنواع أساسية للأزرار التي يمكن إضافتها إلى نموذج :

أزرار الإرسال وإعادة تأهيل النموذج:

- يساعد زر الإرسال في إرسال بيانات النموذج إلى المخدم حتى تجري معالجتها عبر الملف المحدد في الواصفة Action الخاصة بالنموذج وذلك باستخدام الطريقة المحددة في الواصفة Method المرتبطة بالنموذج
- يستخدم هذا النمط الوسم input على أن يتم إسناد القيمة "Submit" إلى الواصفة type
- يمكن استخدام التأشيرة button لتأدية نفس الوظيفة

مثال:

```
<input type="submit" name="test" value="Send">
```

أو

```
<button type="submit" name="test" value="myValue">text on
button</button>
```

- يؤدي زر إعادة تأهيل النموذج Reset إلى إفراغ الحقول من جميع القيم التي تم إدخالها من قبل المستخدم دون إرسال أية بيانات إلى المخدم
- يمكن استخدام التأشيرة input أو التأشيرة button لتحديد زر من هذا النمط بشرط إسناد قيمة "Reset" إلى الواصفة Type

مثال:

```
<input type=”reset” value=”reset”>
```

أو

```
<button type=”reset” >text on reset button </button>
```

عناصر النماذج: الأزرار

الأزرار من نمط صورة:

- يُستخدم هذا النوع من الأزرار بهدف إضفاء جمالية على النموذج عبر تمثيل زر الإرسال بصورة
- يجري استعمال التأشيرة input مع إسناد القيمة "image" إلى الواصفة Type
- يجري تحديد ملف الصورة المُستخدَمة بإسناد المسار النسبي أوالمطلق لملف الصورة على المخدم إلى الواصفة Src

مثال:

```
<input type=”image” src=”./images/myImage.gif”>
```

عناصر النماذج: الأزرار

الأزرار متعددة الأغراض:

- يساعد هذا النمط في إنشاء أزرار ذات أغراض متعددة مختلفة عما أوردناه حتى الآن
- يجري في هذا النمط من الأزرار استخدام الأحداث لتحديد الفعل ورد الفعل
- تستلزم هذه الأزرار استخدام نص برمجي من جهة الزبون
- تستخدم هذه الأزرار التأشيرة Input أو التأشيرة Button بعد إسناد القيمة Button إلى الواصفة type

مثال:

```
<input type="button"
value="Please Don't Press This Button"
onclick="window.alert('any message.')" />
```

أو

```
<button type="button"
value="anyValue"
onclick="window.alert(any message.')"> 'Please do not press
this button again.</button>
```

جرى في المثال السابق الإشارة إلى حدث الضغط على زر، بنص برمجي من جهة الزبون يقوم بإظهار

رسالة. ويكمن الاختلاف الرئيسي بين Button و Input في أن التأشيرة Button تأخذ قيمة النص الموجود على الزر من النص المحصور بالتأشيرة، أما في حالة التأشيرة input فيتحدد النص الموجود على الزر بقيمة الواصفة value الخاصة بهذه التأشيرة.

عناصر النماذج: الحقول المخفية

الحقول المخفية:

- تعد الحقول المخفية من العناصر الهامة في النماذج رغم كونها غير مرئية للمستخدم، حيث يهدف هذا النمط من الحقول إلى تمرير معلومات –عند إرسال النموذج– دون إظهارها في حقل واضح ضمن النموذج

- يُستخدم هذا النمط من الحقول عادة في حالتين:

– لتمرير قيم إلى المخدم ليس لها أي دلالة للمستخدم، حيث يستفيد من هذه القيم النص البرمجي الذي يعالج المعطيات من جهة المخدم

– لتمرير معلومات يجب أن تبقى غير مرئية للمستخدم لأسباب تتعلق بسريتها. إذ يضمن إخفاء بعض المعلومات المُرسلة بهذه الطريقة حمايتها من المستخدم البسيط أو المبتدأ

- يستخدم هذا النمط من الحقول التأشيرة input بعد إسناد القيمة "Hidden" إلى الواصفة Type. أما القيمة التي يجري إرسالها عند إرسال النموذج، فهي القيمة المُسنَدة إلى الواصفة Value

مثال:

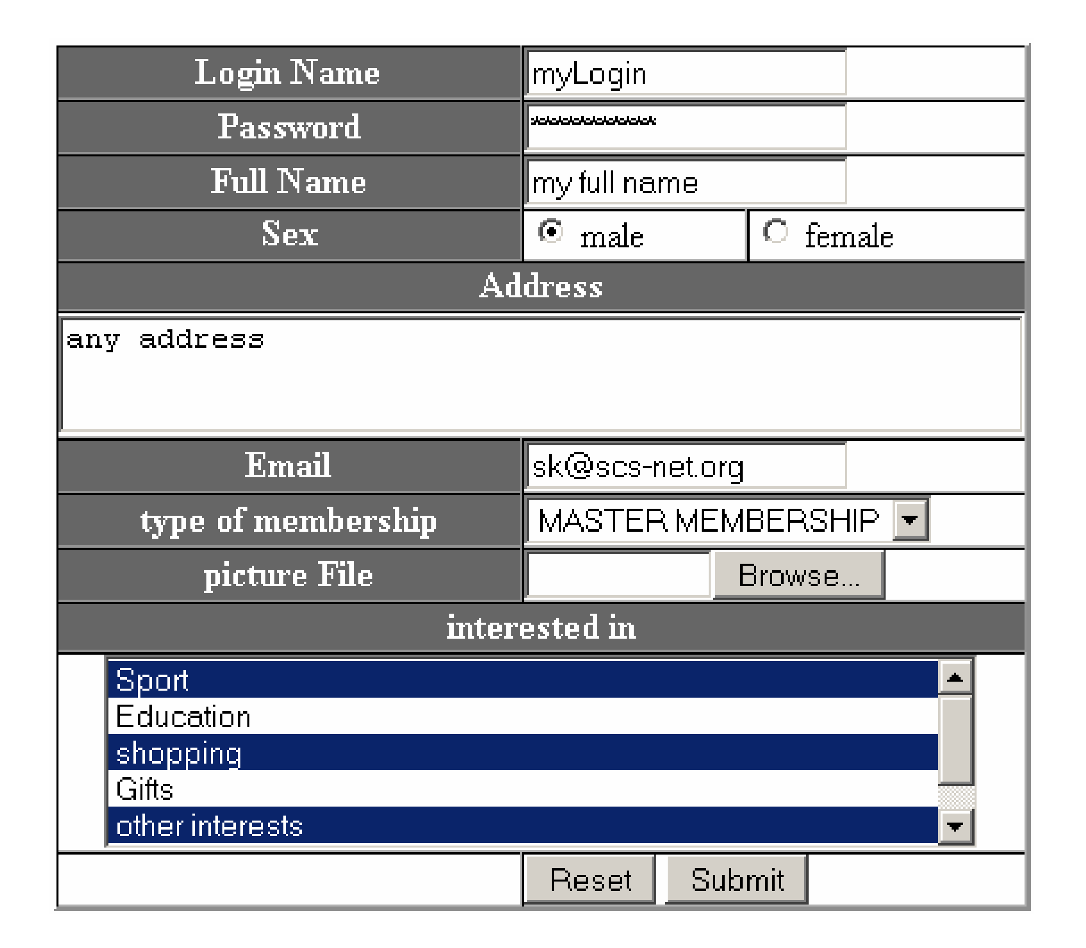
```
<input type="hidden" name="myHidden" value="value we want to hide"
/>
```

تكون أحد القيم التي تصل إلى المخدم، في حال جرى إرسال نموذج يحتوي النص أعلاه، هي القيمة:

myHidden=value we want to hide

تمرين عام

نريد إنشاء نموذج يعمل على جمع المحددة فيما يلي لإرسالها إلى ملف باسم register.aspx. نفترض أن النموذج يمتلك الشكل التالي:



الحل:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Test Form</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1">
<style type="text/css">
<!--
.style1 {color: #FFFFFF}
-->
</style>
</head>

<body>
<form action="register.aspx" method="post" enctype="multipart/form-
data">
<table width="340" border="2" cellpadding="0" cellspacing="0"
bordercolor="#FFFFFF">
<tr>
<th width="185" bgcolor="#666666" scope="row"><span
class="style1">Login Name</span></th>
<td colspan="2"><input name="login" type="text" id="login"></td>
</tr>
```

```
<tr>
<th bgcolor="#666666" scope="row"><span
class="style1">Password</span></th>
<td colspan="2"><input name="pass" type="password" id="pass"></td>
</tr>
<tr>
<th bgcolor="#666666" scope="row"><span class="style1">Full Name
</span></th>
<td colspan="2"><input name="fullName" type="text"
id="fullName"></td>
</tr>
<tr>
<th bgcolor="#666666" scope="row"><span
class="style1">Sex</span></th>
<td width="64"><input name="sex" type="radio" value="male">
male</td>
<td width="81"><input name="sex" type="radio" value="radiobutton">
female</td>
</tr>
<tr>
<th colspan="3" bgcolor="#666666" scope="row"><span
class="style1">Address</span></th>
</tr>
<tr>
<th colspan="3" scope="row"><textarea name="address" cols="50"
id="textarea"></textarea></th>
</tr>
<tr>
<th bgcolor="#666666" scope="row"><span
class="style1">Email</span></th>
<td colspan="2"><input name="email" type="text" id="email"></td>
</tr>
<tr>
<th bgcolor="#666666" scope="row"><span class="style1">type of
membership</span></th>
<td colspan="2"><select name="select">
<option>MASTER MEMBERSHIP</option>
<option>NORMAL MEMBERSHIP</option>
<option>GUEST MEMEBERSHIP</option>
</select></td>
</tr>
<tr>
<th bgcolor="#666666" scope="row"><span class="style1">picture
File</span></th>
<td colspan="2"><input name="myPicture" type="file" id="myPicture"
size="10"></td>
</tr>
<tr>
<th colspan="3" bgcolor="#666666" scope="row"><span
class="style1">interested in </span>            </th>
</tr>
```

```
<tr>
<th colspan="3" scope="row"><select name="select2" size="5"
multiple>
<option>Sport</option>
<option>Education</option>
<option>shopping</option>
<option>Gifts</option>
<option>other interests</option>
<option>___________________________________________________</option
>
</select></th>
</tr>
<tr>
<th scope="row"> </th>
<td colspan="2"><input type="reset" name="Submit2" value="Reset">
<input type="submit" name="Submit" value="Submit"></td>
</tr>
</table>
</form>
</body>
</html>
```

الفصل الثالث – الجزء الأول

عنوان الموضوع:

طلب HTTP باستخدام الطريقتين POST و GET

الكلمات المفتاحية:

طلب، ترويسة، بروتوكول، إجابة/استجابة.

ملخص:

يعتمد بروتوكول HTTP على مجموعة من الطرق لإرسال طلبات HTTP أهمها طريقتي GET و POST. سنغطي في هذه الجلسة بعض المعلومات العامة عن طلب برتوكول HTTP وسنستعرض الفروقات الاساسية في استخدام كلا الطريقتين GET و POST.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- طلبات HTTP
- طرق إرسال الطلبات
- طريقتي GET و POST و الشروط المثلى لاستعمالهما

المخطط:

المجموعات

- 4 وحدات (Learning Objects) (عناوين الـــ LO بالترتيب المحدد)

HTTP POST 1.1.1 والطريقة HTTP GETالطريقة

ذكرنا في جلسة سابقة أثناء استعراضنا للتأشيرة "Form"، الواصفة "Method" وبيّنا إمكانية إسناد إحدى القيمتين "POST" أو "GET" لهذه الواصفة، وذكرنا أن هذه الواصفة تحدد كيف سيجري إرسال البيانات أو الطريقة التي ستُعتمد لإرسال طلب HTTP.

لفهم الفرق بين هذين الطريقتين لا بد لنا من شرح بسيط لأجزاء طلب بروتوكول HTTP .

البروتوكول HTTP:

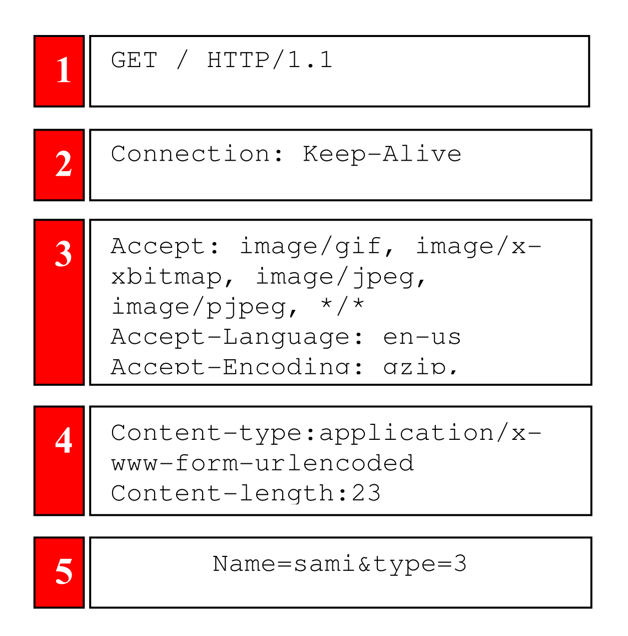
يُعد البروتكول HTTP البروتوكول التطبيقي الأساسي المُستخدم لتناقل البيانات على شبكة الأنترنت.

ويعتمد البروتوكول HTTP آلية (طلب/استجابة) بين التطبيق الزبون (المتصفح) والتطبيق المخدم (مخدم الوب)

طلب HTTP:

تبدأ مناقلة HTTP بإرسال طلب HTTP.

يتكون طلب HTTP من مجموعة من الأجزاء كما يظهر في الشكل التالي. سنستعرض فيما يلي هذه الأجزاء ونحدد لاحقاً نوع المعلومات التي يحملها كل جزء.



الطريقةHTTP GET و HTTP POST

طلبHTTP

1. مُعرِّف (محدد) المصدر / الطريقة
2. الترويسة العامة
3. ترويسة الطلب
4. ترويسة الكيان
5. جسم الكيان

يتألف طلب HTTP مما يلي:

1. يتكون الجزء الأول من طلب http من العبارة (Method URI HTTP/version)، الذي يطلب الصفحة من المخدم عبر مُعرِّف المصدر (URI) باستخدام الطريقة Method. يُمثِّل القسم الباقي من هذا الجزء من الطلب (الجزء HTTP/1.1) نسخة البروتوكول المستخدمة.
2. يحتوي الجزء الثاني من الطلب الترويسة العامة ويحدد معلومات عامة كالتاريخ الحالي أو معلومات

خاصة كالاتصال الحالي. يُعد هذا الجزء غير إجباري بحيث يمكن أن يتم إرسال طلب HTTP من دون هذا الجزء.

3. يحتوي الجزء الثالث الذي يدعى ترويسة الطلب على معلومات تتعلق بالزبون مُرسل الطلب وبنمط البيانات المُرسَلة واسم المضيف. يُعد هذا الجزء غير إجباري بحيث يمكن أن يتم إرسال طلب HTTP من دون هذا الجزء.
4. يُستخدم الجزء الرابع الذي يدعى ترويسة الكيان، عند الشروع بإرساله وهو يحتوي معلومات تتعلق بنمط الكيان، وطوله، ومصدره وطريقة ترميزه. يكون هذا الجزء غير ضروري في حال عدم إرسال أي كيان.
5. يمثل الجزء الأخير جسم الكيان ويحتوي القيم التي يقوم الطلب بإرسالها مثل قيمة حقل ما.

طريقة GET أو POST

بالعودة إلى مخطط الأجزاء المكونة لطلب HTTP، نجد أنه يحتوي على الطريقة التي سيجري استخدامها في إرسال المعلومات. تأخذ هذه الطريقة إحدى القيمتين التاليتين:

GET: يجري طلب الوثيقة المحددة بالمحدد URL، ويجري إرسال بيانات المستخدم إلى العنوان المعين بالمحدد URL ضمن الترويسة نفسها.

POST: يجري طلب الوثيقة المحددة بالمحدد URL، ويجري إرسال بيانات المستخدم إلى العنوان المعين بالمحدد URL مع تضمين البيانات ضمن الجزء الخاص بجسم الكيان وليس في أي من الترويسات.

تُستخدم هاتان الطريقتان أثناء العمل مع نماذج XHTML ، كما يدعم البروتوكول HTTP العديد من الطرق الأخرى نذكر منها:

HEAD: تتقدم بطلب مطابق للطلب الذي تتقدم به GET ولكن الجواب على هذا الطلب يتكون من ترويسة دون أي جسم. تفيد هذه الطريقة في الحصول على المعلومات الموجودة في ترويسة الجواب HTTP دون الحاجة إلى نقل كامل محتوى الجواب بما في ذلك نص الوثيقة التي يجري نقلها عادةً عند الإجابة على الطلب.

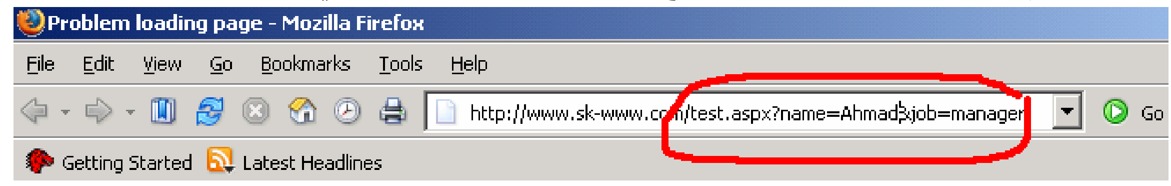
DELETE: تُستخدم لحذف الوثيقة المُشار إليها بالمحدد URL وهي نادرة الاستخدام.

TRACE: عند إرسال الطلب تجري إعادة نفس الطلب كاستجابة. بحيث تُمكن الزبون من معرفة المعلومات التي تضيفها المخدمات الوسيطة إلى الطلب أو معرفة أي تغيير يحصل على محتوى الطلب المرسل.

OPTIONS: تعيد قائمة بطرق HTTP التي يدعمها المخدم ويمكن استخدامها لاختبار عمل مخدم الوب.

GET أو POST

عند استخدام GET لإرسال بيانات النموذج من الضروري مراعاة مايلي:



- يجب ألا يكون حجم هذه البيانات كبيراً
- يجب ألا تحمل البيانات المُرسلة طابع السرّية

تُستخدَم GET أيضاً بكثرة عندما تمرير قيمة معينة لملف عن طريق ارتباط تشعبي:

```
<a href="test.aspx?x=10">click here</a>
```

أما الطريقة POST فتستخدم لإرسال بيانات النموذج في حال:

- أردنا إرسال بيانات بحجم كبير نسبياُ
- إذا كانت المعلومات التي يتعامل معها النموذج حساسة

بعد أن تعرفنا على الفرق بين إرسال طلب HTTP باستخدام GET وإرسال طلب HTTP باستخدام POST، لا بد من توضيح سبب استخدام كلٍ من الطريقتين في نماذج XHTML:

عند استخدام GET لإرسال بيانات النموذج من الضروري مراعاة مايلي:

- يجب ألا يكون حجم هذه البيانات كبيراً نظراً لأنّ حجم البيانات الممكن إرسالها محدود بالحجم الأعظمي لمحدد المصدر، أي الـ URL
- يجب ألا تحمل البيانات المُرسلة طابع السرّية لأن هذه البيانات ستكون مكشوفة وستظهر ضمن حيز عنوان المستعرض

تُستخدَم GET أيضاً بكثرة عندما تمرير قيمة معينة لملف عن طريق ارتباط تشعبي. فعلى سبيل المثال، إذا أردنا تمرير القيمة X=10 إلى الملف test.aspx عند النقر على وصلة ما ولتكن الوصلة click here، تكون

الصيغة على الشكل التالي:

```
<a href="test.aspx?x=10">click here</a>
```

أما الطريقة POST فتستخدم لإرسال بيانات النموذج في حال:

- أردنا إرسال بيانات بحجم كبير نسبياً مثل ملف أو مربع نص يحتوي على حجم كبير من البيانات
- إذا كانت المعلومات التي يتعامل معها النموذج حساسة بحيث لايمكن إظهارها ضمن حيز العنوان في المستعرض

الجزء الثاني

عنوان الموضوع:

مقدمة إلى ASP.NET

الكلمات المفتاحية:

إطار عمل، بيئة تطوير، أدوات تطوير، تثبيت، معالج.

ملخص:

تعد ASP.NET من التقنيات الرائدة في مجال النصوص البرمجية من جهة المخدم. سنتعرف في هذا الفصل على كيفية تثبيت إطار العمل الخاص بهذه التقنية بالإضافة إلى بيئة تطوير Visual Studio.net.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- مقدمة عن ASP.NET
- تثبيت إطار العمل Net.
- تثبيت Visual Studio.Net
- المزايا الرئيسية لتقنية ASP.NET

سينصب تركيزنا فيمايلي على الحديث عن ASP.NET كمثال عن أحد أهم التقنيات المعتمدة في كتابة النصوص البرمجية من جهة المخدم، كما سنتكلم عن ميزات هذه التقنية وعن تفاصيل استخدامها.

تثبيت بيئة تطوير ASP.NET:

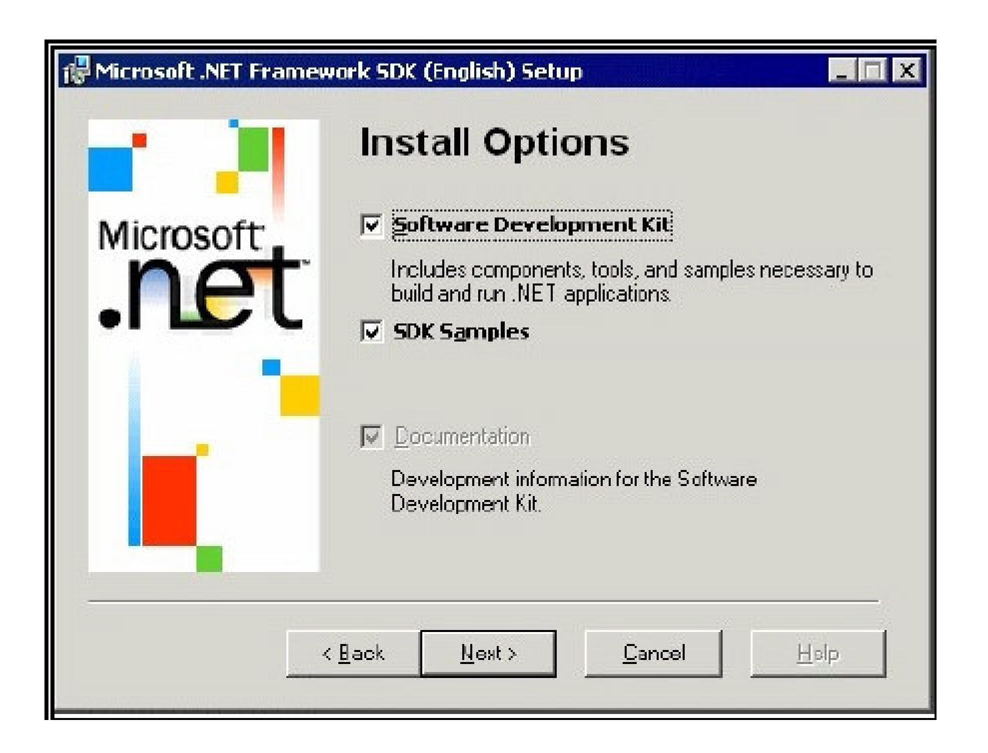
يُعد تثبيت بيئة تطوير ASP.NET سهل نسبياً. إذ تتكون حزمة التثبيت من ملف تنفيذي وحيد يقوم بتثبيت إطار العمل .NET الذي يتضمن ASP.NET. ويقدم معالج التثبيت خيارات تتعلق بإمكانية تثبيت أمثلة تجريبية وملفات توثيق.NET..

يمكن أن يطلب معالج التثبيت تحديث بعض مكونات Windows Installer، لا بد عندها من الموافقة على هذه الرسالة للاستمرار بعملية التثبيت، إذ يمكن أن تظهر شاشة حوار كما يلي:



تشير هذه النافذة إل ضرورة تثبيت مكون MDAC 2.7 الذي يعتبر مكون غير إجباري، ولكن من الأفضل تثبيته لنتمكن من استخدام مزايا مختلفة تتعامل مع أدوات الوصول إلى قواعد البيانات ADO.

عند بدء التثبيت تظهر نافذة معالج التثبيت وهي على الشكل:



يقدم المعالج إمكانية تثبيت أدوات التطوير البرمجية إضافة إلى مجموعة من الأمثلة.

في حال اختيار تثبيت الأمثلة، يجري تثبيت تلقائي لمكون MSDE (محرك قاعدة بيانات مايكروسوفت) مع قاعدة بيانات تجريبية كجزء من الأمثلة.

يبلغ حجم إطار عمل NET. فقط دون أي أمثلة 18ميغا بايت.

تثبيت بيئة تطوير ASP.NET

بيئة Visual Studio .NET

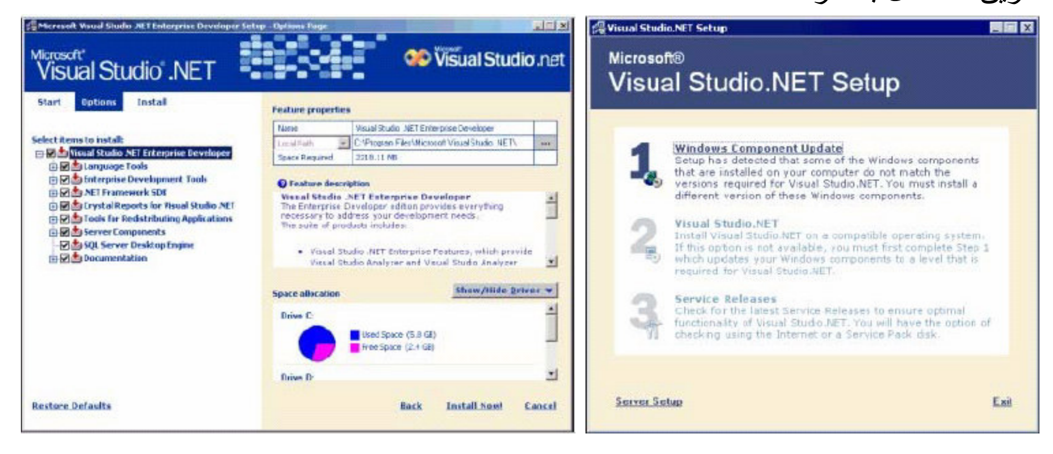
بالرغم من أنّ تثبيت Visual studio .NET ليس شرطاً لازماً لكتابة وتشغيل ملفات ASP.NET، يصعب إهمال بيئة التطوير تلك لما تقدمه من مرونة عالية أثناء عملية التطوير وذلك من خلال واجهات استخدام عناصر التحكم، وآليات إزالة الأخطاء الناجمة عن بيئة التشغيل، وآليات التحسس الذكي لمفردات اللغة، بالإضافة إلى الكثير من الميزات الأخرى.

تثبيت Visual Studio .NET:

- يثبت معالج التثبيت في Visual Studio .NET مجموعة من المكونات التي تتضمن إطار عمل NET.:
  - نسخة MS installer 2.0 أو أي نسخة أحدث؛
  - نسخة MSFront page 2000 web Extention Client؛

o ملفات التشغيل الخاصة بعملية تثبيت البيئة؛
o النسخة 6.00 من مستعرض Internet Explorer بالإضافة إلى Internet Tools؛
o مكونات النسخة 2.7 من MS Data Access؛
o إطار عمل Microsoft .NET.

• ينفذ معالج التثبيت عملية التحقق من التحديثات المتوفرة في بيئة التشغيل بحيث يجري تثبيتها آلياً بعد الحصول عليها عن طريق الاتصال بالانترنت.

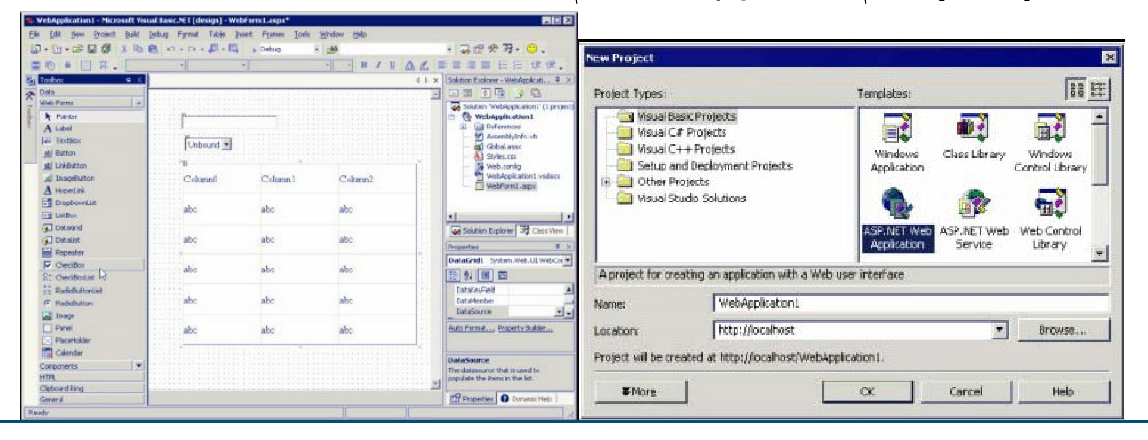


تثبيت Visual Studio.net

نظراً لطبيعة إطار عمل NET.، لن يطلب المعالج تثبيت أدوات خاصة للتعامل مع لغات مختلفة كما هي الحال في النسخ القديمة من بيئة التطوير تلك، إذ يكفي تشغيل Visual studio .NET واختيار اللغة التي سيجري استخدامها في البرمجة، ونوع التطبيق المراد إنشاؤه. يُسهّل هذا الأسلوب على المطوِّر استخدام بيئة العمل مع أي لغة من لغات إطار العمل NET. دون الحاجة إلى التأقلم مع بيئة جديدة للعمل.

استخدام Visual Studio .NET للعمل على ASP.NET:

لتطوير تطبيقات ASP.NET باستخدام Visual studio .NET يكفي اختيار ASP.NET WEB Application من خيارات نافذة New project. تُنشئ هذه العملية موقع وب مع بعض الصفحات التلقائية. من هذه النقطة يمكنك البدء باستخدام آلية اختيار عناصر التحكم المختلفة من واجهة التصميم.



لن ندخل في التفاصيل الخاصة بالعمل على بيئة Visual studio .net وسنركز على تفصيلات عمل تقنية ASP.NET وصيغتها بغض النظر عن بيئة التطوير.

ملاحظة:

لا بد من التنويه أيضاً أن ASP.NET تحتاج، كتقنية نصوص برمجية من جهة المخدم، إلى تثبيت مخدم IIS.

لماذا ASP.NET

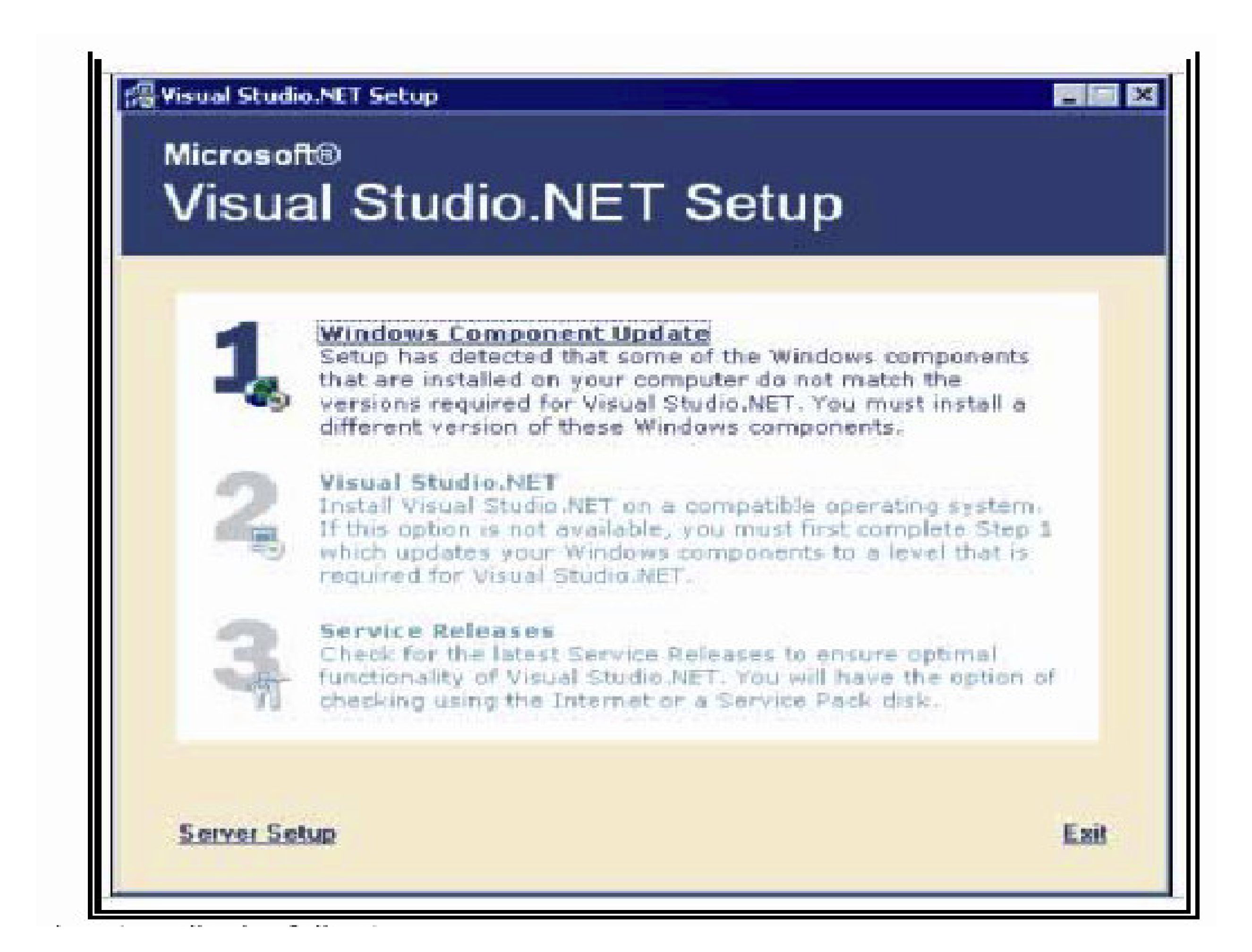
افتقرت نسخة ASP الأولى التي طورتها Microsoft، بالرغم من كونها تجربة مميزة، إلى الكثير من عناصر المرونة. فما الذي يدفعنا لاختيار ASP.NET كبيئة تطويروما هي النقاط الأساسية التي تجعل من هذه اللغة من أنسب اللغات لتطوير التطبيقات على الوب.

- تعتمد لغة ASP.NET على إمكانية استخدام النصوص البرمجية المعالجة أو معالجة النصوص البرمجية ولا تعتمد على استخدام مفسِّر لتفسير النص البرمجي أثناء التنفيذ مما يجعل ASP.NET متميزة من حيث الأداء؛
- تقدم ASP.NET إطار ذو فعالية عالية بأقل حجم ممكن للنص البرمجي؛
- يقدم إطار NET. مجموعة من المكونات الخاصة بالتعامل مع التجهيزات المختلفة بما فيها الأجهزة الخلوية؛
- تقدم ASP.NET من خلال دعمها لتقنيات XML و XHTML بيئة تطوير متوافقة مع المعايير القياسية؛
- جرت إعادة تطوير ASP.NET من الصفر مما ساعد على التخلص من أغلب الأخطاء والمشاكل التي احتوت عليها أسلافها.
- تقدم ASP.NET البيئة المناسبة لكتابة نص برمجي نظيف سهل الفهم.
- تحسن ASP.NET قابلية النقل، والتوسع، والأمان، والموثوقية.
- تدعم إطار عمل NET. العديد من لغات البرمجة مثل #C ، #J، VB.NET مما يجعل من الممكن تطوير عناصر تحكم أو مكونات بأي لغة من هذه اللغات وإعادة استخدامها ضمن ASP.NET.

نشاط

نفذ عملية تثبيت Net Framework. (يمكن لكل طالب أن ينفذها على جهازه أو أن ينفذها المشرف لمرة واحدة أمام الجميع) اعتماداً على مراحل التثبيت التالية.

- **ظهور نافذة التثبيت/التنصيب الأولى:**

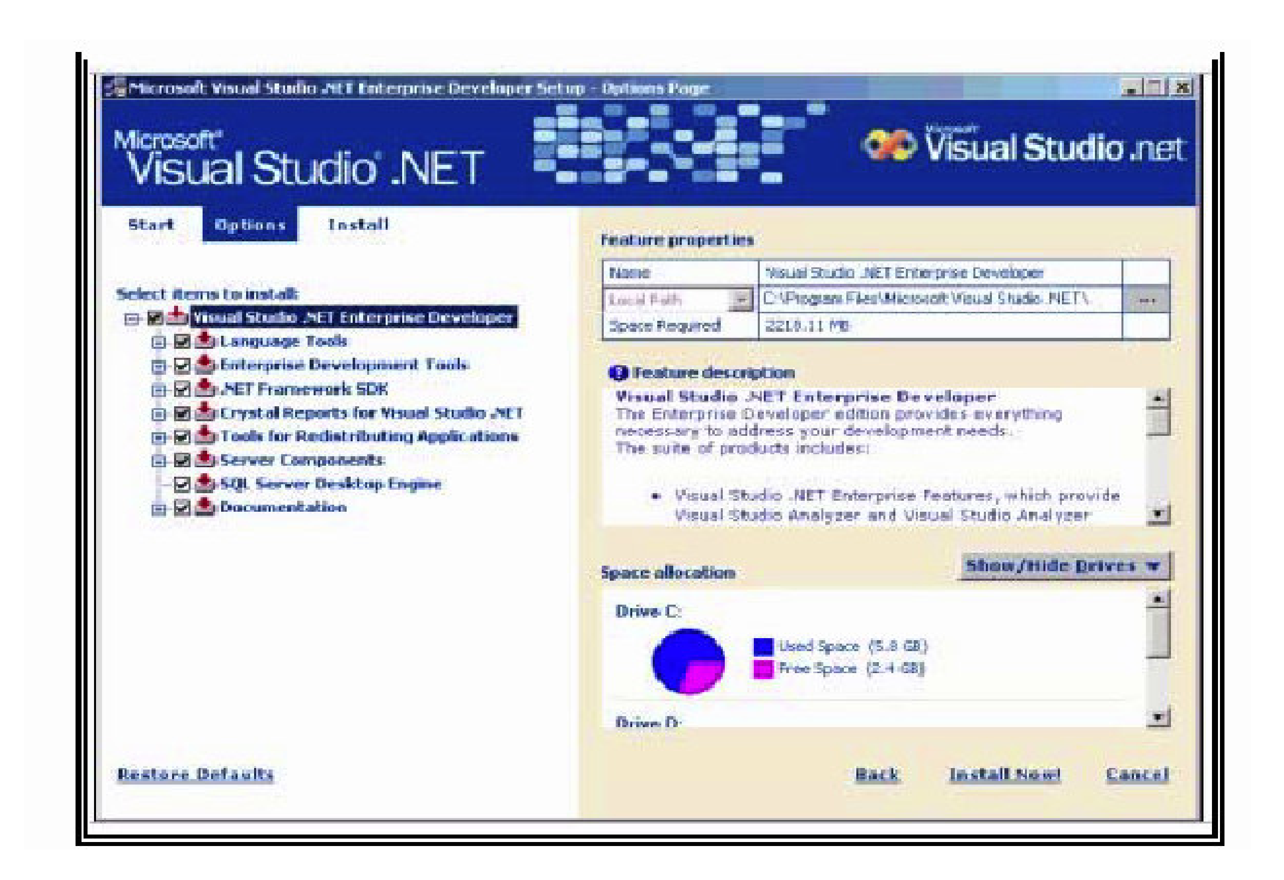


- يساعد برنامج التحديث على تثبيت مايلي:

- Windows 2000 Service Pack 2, if installing on Windows 2000.
- Microsoft Windows Installer 2.0
- Microsoft FrontPage 2000 Web Extensions Client
- Setup Runtime Files
- Microsoft Internet Explorer 6.0 and Internet Tools (this requires a reboot)
- Microsoft Data Access Components 2.7
- Microsoft .NET Framework

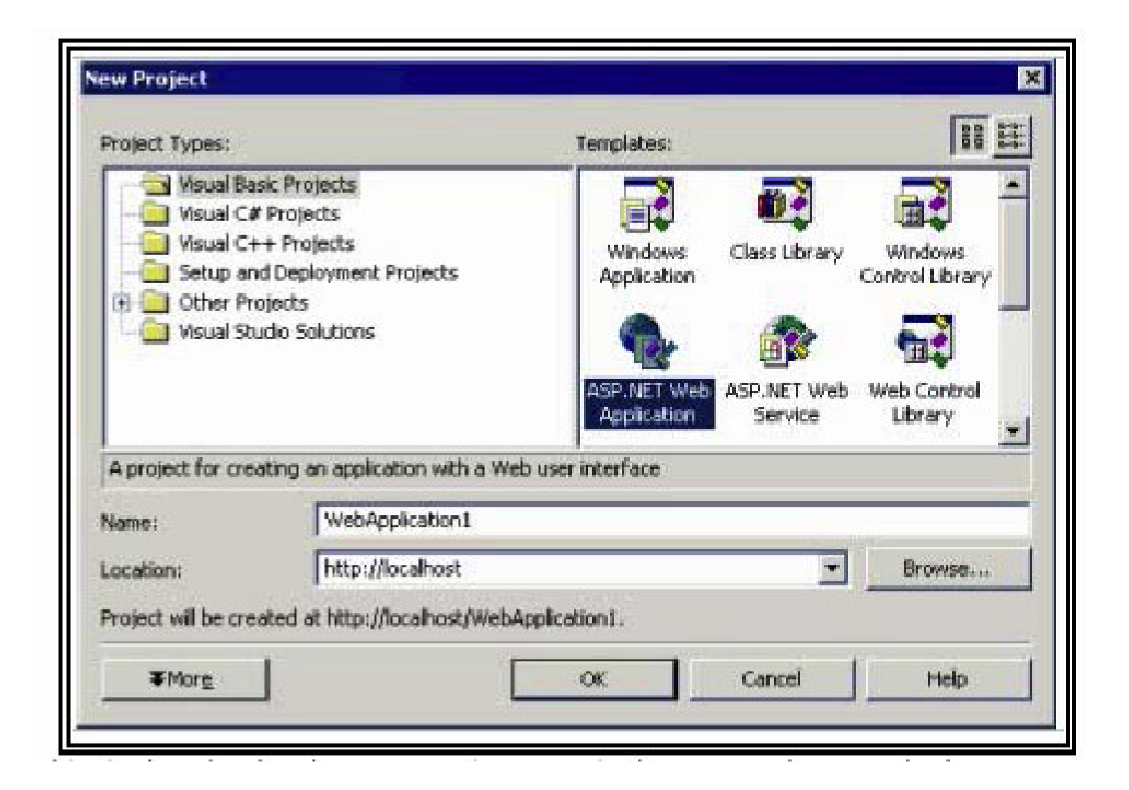
- **يمكن إدخال اسم مستخدم وكلمة مرور بحيث يجري استخدامهما عند إعادة الإقلاع أوتوماتيكياً مما يساعد على تنفيذ عملية التثبيت آلياً دون الحاجة لتدخل المستخدم.**

- **لتثبيت Visual Studio .Net نتبع نفس خطوات التثبيت التي كانت مُتَّبعة في النسخ السابقة:**

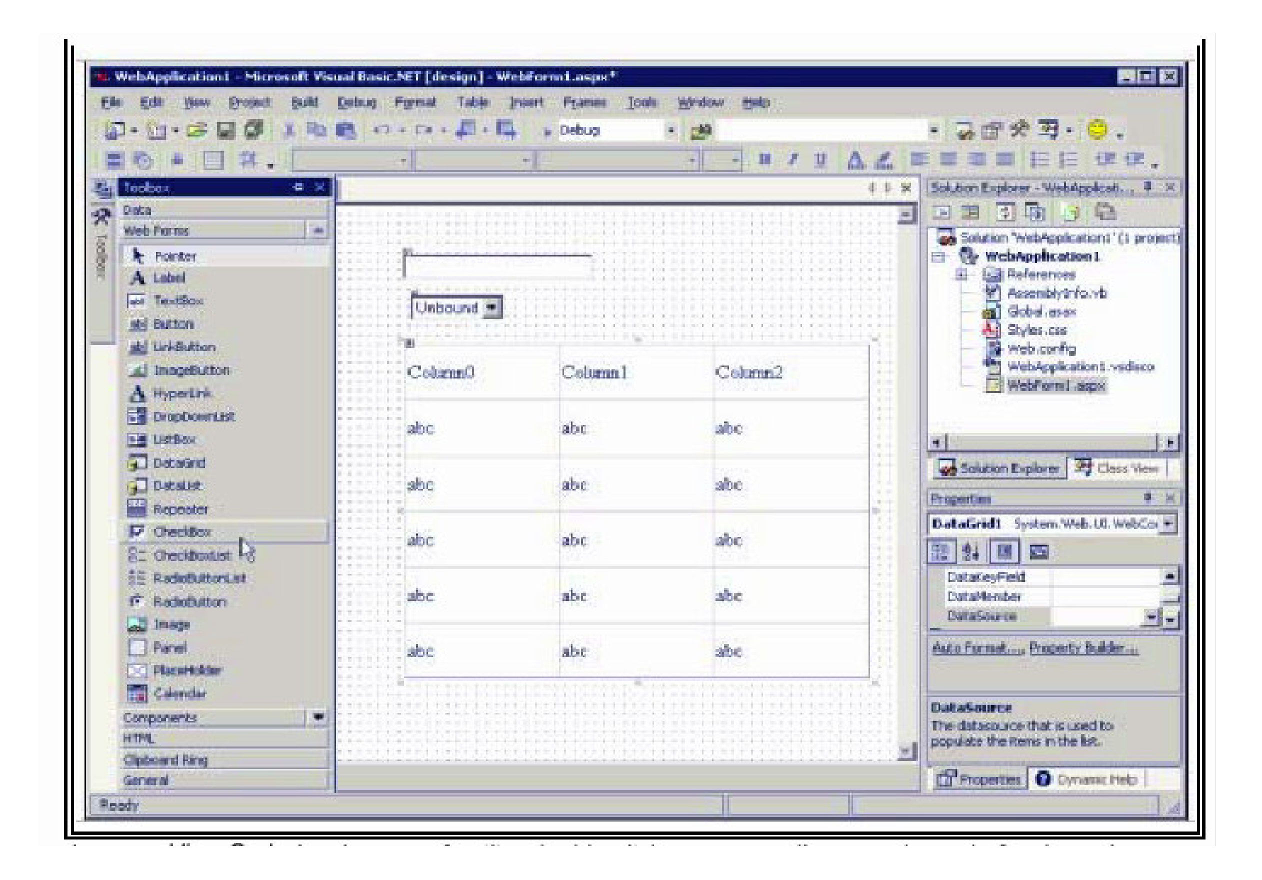


- عند الانتهاء من الخطوة السابقة، يُترك للمستخدم خيار جعل المُنتَج يعتد التحديث الآلي عبر الإنترنت عندما تظهر تحديثات جديدة.

- لا تعتمد النسخة الحديثة على جعل المُستخدم يختار اللغة ومن ثم يُشغِّل الأداة الموافقة، وإنما يجري الآن إقلاع Visual Studio .Net واختيار اللغة والتطبيق المُراد توليده مباشرةً.



- **توليد تطبيقات ASP.Net في Visual Studio.Net:**
  - الذهاب إلى New Project dialog (الظاهر في الشكل أعلاه)
  - اختيار ASP.Net Web Application
  - يجري بناء موقع وب مع عدد من الصفحات التلقائية
  - يمكن الآن اختيار أدوات التحكم وعناصر صفحات الوب من الواجهات المُتاحة فيما يلي:



- **عمليات تثبيت أخرى:**
  - **ODBC .NET Data Provider**: الذي يوفر إمكانية الوصول إلى سواقات ODBC.
  - **Mobile Internet Toolkit**: الذي يوفر إمكانية تطوير مواقع تدعم تجهيزات نقالة (هواتف، PDA).
  - **Internet Explorer Web Controls**: أدوات مُساعدة خاصة بالمتصفح.

عنوان الموضوع:

عناصر تحكم HTML من جهة المخدم

الكلمات المفتاحية:

عنصر تحكم، جهة المخدم ، وب، تأشيرة

ملخص:

قدمت ASP.NET تقنيات مميزة تحقق تحكم أكبر بمكونات صفحة الوب. تعتبر عناصر التحكم من جهة المخدم من أهم تلك التقنيات.

سنحاول خلال هذه الجلسة التعرف على جزء من هذه العناصر وهو عناصر تحكم HTML

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- عناصر التحكم من جهة المخدم
- تفاصيل حول عناصر تحكم HTML

الإقلاع مع ASP.NET

اطَّلعنا في الجلسات الماضية على مفهوم النصوص البرمجية من جهة المخدم وسنحاول من خلال الجلسات التالية تغطية تقنية ASP.NET على اعتبارها إحدى التقنيات الرائدة في هذا المجال.

إعداد البيئة:

لا بد لنا –حتى نتمكن من تشغيل النصوص البرمجية من جهة المخدم– من وضع ملفات النصوص البرمجية على المخدم ضمن المجلد wwwroot أو ضمن أحد المجلدات الافتراضية.

لا بد أن نتأكد أيضاً من:

- تثبيت إطار العملNET.
- تثبيت مخدم وب IIS
- التأكد من إعطاء صلاحية التنفيذ للملفات ضمن المجلد الحاوي على ملفات ASPX

مثال: صفحة ASP.NET بسيطة جداً

لنبدأ أولاً بصفحة ASP.NET لا تحتوي أي أوامر ASP.net اي تحوي فقط تعليمات HTML:

```
<html>
<head>
</head>
<body>
<p> this is only a test that contain html.</p>
</body>
</html>
```

يمكننا أن نحفظ هذا النص بلاحقة (ASPX.) ثم نقوم بتشغيله في المستعرض والوصول إليه باستخدام محدد المصدر القياسي الخاص به.

لا يحوي النص في هذا المثال أي تعليمات ASP.net لتنفيذها وسيقوم هذا الملف بإظهار محتوى الصفحة على المستعرض.

اطّلعنا في الجلسات الماضية على مفهوم النصوص البرمجية من جهة المخدم وسنحاول من خلال الجلسات التالية تغطية تقنية ASP.NET على اعتبارها أحد التقنيات الرائدة في هذا المجال.

إعداد البيئة:

لا بد لنا –حتى نتمكن من تشغيل النصوص البرمجية من جهة المخدم– من وضع ملفات النصوص البرمجية على المخدم ضمن المجلد wwwroot أو ضمن أحد المجلدات الافتراضية.

لا بد أن نتأكد أيضاً من:

- تثبيت إطار العمل NET.
- تثبيت مخدم وب IIS
- التأكد من إعطاء صلاحية التنفيذ للملفات ضمن المجلد الحاوي على ملفات ASPX

استخدام النصوص البرمجية فيASP.NET

لاستخدام نصوص ASP.net ضمن ملف ASPX لدينا مجموعة من الخيارات المتاحة:

- استخدام التأشيرة <% %> لحصر النص البرمجي الخاص ب ASP.NET. حيث جرى اعتماده لضمان التوافقية مع نصوص ASP (سلف ASP.NET) وله الشكل التالي:

```
<% the code goes here %>
```

- استخدام التأشيرة <Script> الخاص ب HTML مع إسناد القيمة server إلى الواصفة runat بحيث يصبح التركيب على الشكل التالي:

```
<script runat=server>
The code goes here
</script>
```

مشكلةASP

عانت نسخة ASP، من نفس معاناة الكثير من تقنيات النصوص البرمجية من جهة المخدم، والتي تمثلت في عدم إمكانية فصل النص البرمجي عن سياق خرج HTML، لذا كان لا بد دائماً من وضع النص البرمجي في المكان الذي نريد إظهار الخرج فيه.

مثال:

```
<html>
<body bgcolor="yellow">
<center>
<h2>Hello W3Schools!</h2>
<p><%Response.Write(now())%></p>
</center>
</body>
</html>
```

نلاحظ أننا اضطررنا في هذا المثال إلى وضع النص البرمجي في النقطة التي نود إظهار الخرج فيها وهو في حالتنا الوقت والتاريخ.

عانت نسخة ASP، من نفس معاناة الكثير من تقنيات النصوص البرمجية من جهة المخدم، والتي تمثلت في عدم إمكانية فصل النص البرمجي عن سياق خرج HTML، لذا كان لا بد دائماً من وضع النص البرمجي في المكان الذي نريد إظهار الخرج فيه.

ما الإضافة التي قدمتها ASP.NET ؟

قدمت ASP.NET حلاً لمشكلة عدم إمكانية فصل النص البرمجي عن سياق الخرج، باعتمادها تقنية عناصر التحكم من طرف المخدِّم، أو ما يسمى بتقنية Server Control.

عناصر التحكم من جهة المخدم هي بالتعريف تأشيرات يمكن للمخدم فهمها والتعامل معها.

تقدم ASP.NET ثلاثة أنواع أساسية من عناصر التحكم من جهة المخدم:

1. عناصر تحكم HTML
2. عناصر تحكم الوب (و هي أنواع جديدة من الوسوم خاصة بASP.NET)
3. عناصر تحكم تساعد في التحقق من دخل المستخدم

سنستعرض لاحقاً أثناء هذه الجلسة وخلال الجلسات القادمة أهم عناصر التحكم من جهة المخدم وسنوضحها بأمثلة. ولكن قبل البدء بهذا السرد لابد من استعراض ولو بصورة سريعة مفهوم الأحداث في ASP.NET لأننا سنستخدمه في أغلب أمثلتنا.
تعتمد الكثير من التقنيات البرمجية على ما يسمى **البرمجة المقادة بالأحداث**، وهي تعني أن أي نص برمجي يجري تنفيذه يجب أن يكون مرتبطاً بحدث ما ويسمى هذا النص، النص البرمجي الخاص بمعالج الحدث.

لذلك يتوجب على المبرمج، أثناء برمجته لتطبيقات الوب باستخدام لغة ASP.NET، الانتباه إلى ضرورة وضع النصوص البرمجية ضمن التركيبة المناسبة لتقنية البرمجة المقادة بالأحداث والتي تعتمد على ربط الحدث مباشرةً بمعالج الحدث.

الأحداث في ASP.NET

مثال:

لننظر إلى النص البرمجي التالي:

```
<%
lbl1.Text="The date and time is " & now()
%>
<html>
<body>
<form runat="server">
<h3><asp:label id="lbl1" runat="server" /></h3>
</form>
</body>
</html>
```

إذا أردنا طرح السؤال:

متى سيتم تنفيذ هذا النص البرمجي المحدد أعلاه ؟

ستكون الإجابة حتماً : لانعلم!!!

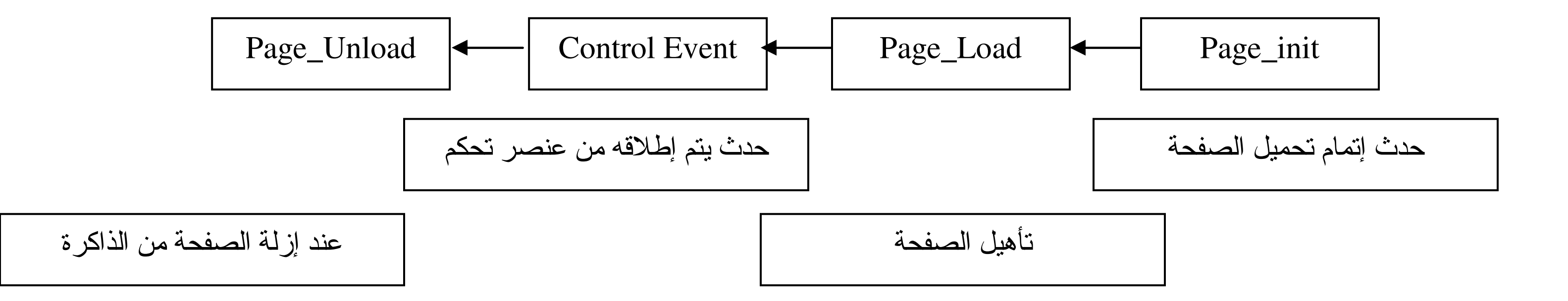
تعتمد الكثير من التقنيات البرمجية على ما يسمى **البرمجة المقادة بالأحداث**، وهي تعني أن أي نص برمجي يجري تنفيذه يجب أن يكون مرتبطاً بحدث ما ويسمى هذا النص، النص البرمجي الخاص بمعالج الحدث.

لذلك يتوجب على المبرمج، أثناء برمجته لتطبيقات الوب باستخدام لغة ASP.NET، الانتباه إلى ضرورة وضع النصوص البرمجية ضمن التركيبة المناسبة لتقنية البرمجة المقادة بالأحداث والتي تعتمد على:
الحدث ← تنفيذ النص البرمجي الخاص بمعالج الحدث.

تعتمد الكثير من التقنيات البرمجية على ما يسمى **البرمجة المقادة بالأحداث**، وهي تعني أن أي نص برمجي يجري تنفيذه يجب أن يكون مرتبطاً بحدث ما ويسمى هذا النص، النص البرمجي الخاص بمعالج الحدث.

لذلك يتوجب على المبرمج، أثناء برمجته لتطبيقات الوب باستخدام لغة ASP.NET، الانتباه إلى ضرورة وضع النصوص البرمجية ضمن التركيبة المناسبة لتقنية البرمجة المقادة بالأحداث والتي تعتمد على ربط الحدث مباشرةً بمعالج الحدث.

أهم الأحداث التي تمر بها الصفحة من بداية عملها



تأخذ معالجات الأحداث في ASP.NET شكل الإجراءات الفرعية ويُعبَر عنها بالصيغة:

```
Sub event_name
Event_handling _code
End sub
```

عناصر تحكم HTML من جهة المخدم

تنحصر مسؤولية مخدم الوب في حالة تأشيرة html التقليدية التي تظهر ضمن ملف ASP.NET بتمرير هذه التأشيرة كنص عادي ليجري تفسيرها من قبل المستعرض ثم إظهار الناتج كصفحة وب. يختلف الحال مع عناصر تحكم HTML من جهة المخدم حيث تُعامل تلك العناصر كمكونات قابلة للبرمجة يفهمها المخدم.

لجعل تأشيرة HTML تُعامَل كعنصر تحكم، يكفي إضافة الواصفة runat لهذه التأشيرة وإسناد القيمة "SERVER" إليها، وإعطاء قيمة للواصفة (ID) بحيث يجري استخدامها لاحقاً من قبل المخدم كمُعرِّف للوصول إلى عنصر التحكم السابق عند تشغيل النص البرمجي.

ملاحظة هامة:

يجب أن تظهر جميع عناصر التحكم من جهة المخدم ضمن التأشيرة <form> مع الانتباه إلى ضرورة اعتبار هذه التأشيرة عنصر تحكم، أي تزويدها بقيمة للواصفة runat وقيمة للواصفة ID، وذلك لإعلام المخدم أن هذا النموذج يجب التعامل معه كنموذج من طرف المخدم، وأنّ العناصر المحصورة بهذا الوسم يمكن الوصول إليها من قبل النص البرمجي من جهة المخدم.

مثال:

في هذا المثال قمنا بالتصريح عن عنصر تحكم من نوع ارتباط تشعبي ضمن ملف ASPX

```
<html>
<body>
<form runat="server">
<a id="link1" runat="server">Visit sk website!</a>
</form>
</body>
```

تم إعطاء القيمة "link1" للواصفة ID الخاصة بهذا العنصر كما يظهر في النص البرمجي.

بتمرير هذه ASP.NET التقليدية التي تظهر ضمن ملف htmlتنحصر مسؤولية مخدم الوب في حالة تأشيرة التأشيرة كنص عادي ليجري تفسيرها من قبل المستعرض ثم إظهار الناتج كصفحة وب. يختلف الحال مع من جهة المخدم حيث تُعامل تلك العناصر كمكونات قابلة للبرمجة يفهمها المخدم.HTMLعناصر تحكم

لجعل تأشيرة HTML تُعامَل كعنصر تحكم، يكفي إضافة الواصفة runat لهذه التأشيرة وإسناد القيمة "SERVER" إليها، وإعطاء قيمة للواصفة (ID) بحيث يجري استخدامها لاحقاً من قبل المخدم كمُعرِّف للوصول إلى عنصر التحكم السابق عند تشغيل النص البرمجي.

من جهة المخدمHTMLعناصر تحكم

بعد أن أوضحنا المبدأ الأساسي لعناصر تحكم HTML من جهة المخدم، سنقوم باستعراض أهم هذه العناصر وأكثرها استخداماً محاولين أثناء هذا السرد توضيح بعض الآليات التي تستخدم مع هذه العناصر من خلال الأمثلة.

عنصر تحكم الارتباط التشعبي HTMLANCHOR:

يعمل عنصر التحكم هذا، لإنشاء ارتباط تشعبي وهو يستخدم التأشيرة <a>.

من أهم معاملات الغرض البرمجي الذي يمثل هذا العنصر :

تعيد قائمة بجميع واصفات هذا العنصر وقيمها	**ATTRIBUTES**
قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هيFALSE	**DISABLED**

محدّد المصدر القياسي للارتباط التشعبي	**HERF**
مُعرّف وحيد لهذا العنصر	**ID**
يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصرHTML وهو لايقوم بتحويل المحارف الخاصة إلى	**INNERHTML**
يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصرHTML وهو يقوم بتحويل المحارف الخاصة إلى	**INNERTEXT**
اسم الارتباط	**NAME**
التي سيجري تطبيقها على هذا العنصرCSS يعيد خصائص	**STYLE**
اسم النافذة الهدف التي سيجري فتح محتوى الارتباط التشعبي فيها	**TARGET**
العنوان الذي سوف يظهر عند تحريك المؤشر فوق هذا الارتباط	**TITLE**
قيمة منطقية تحدد كون هذا العنصر مرئي أو لا	**VISIBLE**
اسم التابع الذي سيجري تنفيذه عند النقر على هذه الوصلة	**ONSERVERC LICK**

مثال:

```
<script  runat="server">
Sub Page_Load
   link1.HRef="http://www.google.com"
   link1.Target="_blank"
   link1.Title="google"

   link2.HRef="http://www.microsoft.com"
   link2.Target="_blank"
   link2.Title="Microsoft"
End Sub
</script>

<html>
<body>

<form runat="server">
<a id="link1" runat="server">Visit google!</a>
<br />
```

```
</form>

</body>
</html>
```

سيقوم مثالنا بإنشاء وصلتين أحدهما لموقع MSN والثاني لموقع google .

كما نلاحظ أنَ كتلة النص البرمجي تم فصلها عن سياق كتلة محتوى الصفحة حيث تم وضع التعليمات كجسم لمعالج الحدث Page_load في بداية المثال.

تبدو بصورة واضحة آلية تعيين قيم الخصائص لعناصر التحكم من جهة المخدم.

عناصر التحكم HTML

عنصر التحكم HTMLBUTTON

يعمل عنصر التحكم هذا على إنشاء زر وهو يستخدم التأشيرة <button>. من أهم معاملات الغرض الذي يمثل هذا العنصر:

ATTRIBUTES	تعيد قائمة بجميع واصفات هذا العنصر وقيمها
DISABLED	قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعّل وقيمتها الافتراضية هي FALSE
ID	مُعرّف وحيد لهذا العنصر
INNERHTML	يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو لايقوم بتحويل المحارف الخاصة إلى HTML
INNERTEXT	يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو يقوم بتحويل المحارف الخاصة إلى HTML
ONSERVERCLICK	اسم التابع الذي سيتم تنفيذه عند النقر على هذه الوصلة
RUNAT	ويحدد أن عنصر التحكم من جهة المخدم يأخذ القيمة Server
STYLE	يعيد خصائص CSS التي سيجري تطبيقها على هذا العنصر

VISIBLE	قيمة منطقية تحدد كون هذا العنصر مرئي أو لا

مثال:

سيُظهِر هذا المثال زرين بلونيين مختلفين وسيُظهِر تعليق مختلف عند نقر كل زر من الزرين.

نلاحظ أننا استخدمنا الأحداث التي تطلقها عناصر التحكم ولم نستخدم حدث تحميل الصفحة Page_load:

```
Sub event_name (source as object , e as eventargs)
…..
End sub
```

يكفي لربط حدث ما بمعالج حدث إعطاء قيمة اسم الحدث للواصفة onserverclick (في حالة مثالنا).

```
<script  runat="server">
Sub button1(Source As Object, e As EventArgs)
  p1.InnerHtml="You clicked the blue button!"
End Sub
Sub button2(Source As Object, e As EventArgs)
  p1.InnerHtml="You clicked the pink button!"
End Sub
</script>

<html>
<body>

<form runat="server">
<button id="b1" OnServerClick="button1"
style="background-color:#e6e6fa;
height=25;width:100" runat="server">
Blue button!
</button>
<button id="b2"
OnServerClick="button2"
style="background-color:#fff0f5;
height=25;width:100" runat="server">
Pink button!
</button>
<p id="p1" runat="server" />
</form>

</body>
</html>
```

عنصر التحكم HTMLFORM

يعمل عنصر التحكم هذا على إنشاء نموذج من جهة المخدم وهو يستخدم التأشيرة <Form>.من أهم

المعاملات الخاصة بالغرض الذي يمثل هذا العنصر:

ACTION	محدد المصدر القياسي للصفحة التي سيتم إرسال البيانات إليها. يجب في حالة عناصر التحكم من جهة المخدم إعطاء نفس محدد الصفحة الحالية دائماً.
ATTRIBUTES	تعيد قائمة بجميع الواصفات وقيمها.
DISABLED	قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هي FALSE
ENCTYPE	تحدد نمط تشفير البيانات حسب أنماط MIME.
ID	مُعرّف وحيد لهذا العنصر
INNERHTML	يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو لايقوم بتحويل المحارف الخاصة إلى HTML
INNERTEXT	يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو يقوم بتحويل المحارف الخاصة إلى HTML
METHOD	تحدد طريقة إرسال البيانات: GET أو POST
NAME	اسم الارتباط
RUNAT	ويحدد أن عنصر التحكم من جهة المخدم يأخذ القيمة Server
STYLE	يعيد خصائص CSS التي سيتم تطبيقها على هذا العنصر
TARGET	اسم النافذة الهدف التي سيتم فتح محتوى الارتباط التشعبي فيها
VISIBLE	قيمة منطقية تحدد كون هذا العنصر مرئي أو لا

عنصر التحكم HTMLGENERIC

عنصر التحكم هذا يمكن استخدامه للتخاطب مع أي تأشيرة من جهة المخدم. من أهم معاملات الغرض الممثل لهذا العنصر

ATTRIBUTES	تعيد قائمة بجميع واصفات هذا العنصر وقيمها
DISABLED	قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هيFALSE
ID	مُعرّف وحيد لهذا العنصر
INNERHTML	يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصرHTML وهو لايقوم بتحويل المحارف الخاصة إلى
INNERTEXT	يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصرHTML وهو يقوم بتحويل المحارف الخاصة إلى
RUNAT	ويحدد أن عنصر التحكم من جهة المخدم Serverيأخذ القيمة
STYLE	التي سيتم تطبيقها على هذا العنصر CSSيعيد خصائص
VISIBLE	قيمة منطقية تحدد كون هذا العنصر مرئي أو لا

مثال:

نتعامل في المثال التالي مع التأشيرة <P> على أنها عنصر تحكم من جهة المستخدم، وقد حددنا قيمة الــ ID الخاص بهذا العنصر ثم جعلنا معالج حدث النقر على زر Submit يغير قيمة innerHtml:

```
<script  runat="server">
Sub submit(sender As Object, e as EventArgs)
if name.value<>"" then
  p1.InnerHtml="Welcome " & name.value & "!"
end if
End Sub
</script>

<html>
<body>

<form runat="server">
Enter your name: <input id="name" type="text" size="30" runat="server" />
<br /><br />
<input type="submit" value="Submit" OnServerClick="submit" runat="server" />
<p id="p1" runat="server" />
</form>

</body>
</html>
```

عنصر التحكم HTMLIMAGE:

يعمل عنصر التحكم هذا على إنشاء عنصر تحكم من جهة المخدم خاص بصورة وهو يستخدم التأشيرة <img>. من أهم معاملات الغرض الذي يمثل هذا العنصر :

تحدد الخاصية محاذاة الصورة نسبة إلى الكتل التي تحيط بها ويمكن أن تأخذ القيمة top ,middle ,bottom ,left ,right	**ALIGN**
وصف بسيط للصورة	**ALT**
تعيد قائمة بجميع واصفات هذا العنصر وقيمها	**ATTRIBUTES**
عرض الإطار.	**BORDER**
قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هي FALSE	**DISABLED**
ارتفاع الصورة	**HEIGHT**
مُعرّف وحيد لهذا العنصر	**ID**
ويحدد أن عنصر التحكم من جهة المخدم يأخذ القيمة Server	**RUNAT**
محدد المصدر القياسي للصورة المراد إظهارها	**SRC**
يعيد خصائص CSS التي سيتم تطبيقها على هذا العنصر	**STYLE**
قيمة منطقية تحدد كون هذا العنصر مرئي أو لا	**VISIBLE**
عرض الصورة	**WIDTH**

مثال:

```
<script  runat="server">
Sub choose_image(Sender As Object, e As EventArgs)
  image1.Src = select1.Value
End Sub
</script>

<html>
<body>

<form runat="server">
<select id="select1" runat="server">
  <option value="smiley.gif">Smiley</option>
  <option value="angry.gif">Angry</option>
  <option value="stickman.gif">Stickman</option>
</select>
<input type="submit" runat="server" value="Display image"
OnServerClick="choose_image">
```

```
<img id="image1" src="smiley.gif" runat="server" width="32" height="32" />
</form>

</body>
</html>
```

في هذا المثال سيجري تعديل الصورة التي يظهرها عنصر التحكم image1 بحسب القيمة المحددة لعنصر التحكم <option> .

عنصر التحكم HTMLInput:

ATTRIBUTES	تعيد قائمة بجميع واصفات هذا العنصر وقيمها
DISABLED	قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هي FALSE
ID	مُعرّف وحيد لهذا العنصر
NAME	اسم الارتباط
ONSERVERCLICK	اسم التابع الذي سيتم تنفيذه عند النقر على هذه الوصلة
RUNAT	ويحدد أن عنصر التحكم من جهة المخدم يأخذ القيمة Server
STYLE	يعيد خصائص CSS التي سيتم تطبيقها على هذا العنصر
TYPE	نمط العنصر
VALUE	قيمة العنصر
VISIBLE	قيمة منطقية تحدد كون هذا العنصر مرئي أو لا

مثال:

```
<script  runat="server">
Sub submit(sender As Object, e as EventArgs)
if name.value<>"" then
  p1.InnerHtml="Welcome " & name.value & "!"
end if
End Sub
</script>

<html>
<body>

<form runat="server">
Enter your name: <input id="name" type="text" size="30" runat="server" />
<br /><br />
<input type="submit" value="Submit" OnServerClick="submit" runat="server" />
<p id="p1" runat="server" />
</form>
```

```
</body>
</html>
```

استخدمنا في هذا المثال الإجرائية الفرعية submit التي ستقوم باختبار القيمة Value للحقل النصي name. نلاحظ أننا قمنا باحتواء جميع عناصر تحكم html ضمن الوسم <form>.

1.4.14عنصر التحكم HTMLINPUTCHECKBOX:

ATTRIBUTES	تعيد قائمة بجميع واصفات هذا العنصر وقيمها
CHECKED	قيمة منطقية تحدد كون العنصر قد تم اختياره أم لا
DISABLED	قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هيFALSE
ID	مُعرّف وحيد لهذا العنصر
NAME	اسم الارتباط
RUNAT	ويحدد أن عنصر التحكم من جهة المخدم Serverيأخذ القيمة
STYLE	يعيد خصائص CSS التي سيتم تطبيقها على هذا العنصر
TYPE	نوع العنصر
VALUE	قيمة العنصر
VISIBLE	قيمة منطقية تحدد كون هذا العنصر مرئي أو لا

مثال:

```
<script  runat="server">
Sub submit(Source As Object, e As EventArgs)
if red.Checked=True then
  p1.InnerHtml="You prefer red!"
else
  p1.InnerHtml="You prefer blue!"
end if
red.checked=false
blue.checked=false
End Sub
</script>

<html>
<body>
<form runat="server">
What color do you prefer?
<br />
<input id="red" type="checkbox" runat="server" /> Red
<br />
<input id="blue" type="checkbox" runat="server" /> Blue
<br />
<input type="button" value="Submit" OnServerClick="submit" runat="server"/>
<p id="p1" runat="server" />
```

```
</body>
</html>
```

عنصر التحكم HTMLINPUTHIDDEN

ATTRIBUTES	تعيد قائمة بجميع واصفات هذا العنصر وقيمها
DISABLED	قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هي FALSE
ID	مُعرّف وحيد لهذا العنصر
NAME	**اسم الارتباط**
RUNAT	ويحدد أن عنصر التحكم من جهة المخدم Server يأخذ القيمة
STYLE	**يعيد خصائص CSS التي سيتم تطبيقها على هذا العنصر**
TYPE	**نوع العنصر**
VALUE	**قيمة العنصر**
VISIBLE	**قيمة منطقية تحدد كون هذا العنصر مرئي أو لا**

مثال:

```
<script  runat="server">
Sub submit(Source As Object, e As EventArgs)
    hidden1.Value=string1.Value
    p1.InnerHtml="Hidden value= " + hidden1.Value
End Sub
</script>

<html>
<body>

<form runat="server">
Enter some text: <input id="string1" type="text" size="25" runat="server" />
<input type="submit" value="Submit" OnServerClick="submit" runat="server" />
<input id="hidden1" type="hidden" runat="server" />
<p id="p1" runat="server" />
</form>

</body>
</html>
```

عنصر التحكم HTMLINPUTIMAGE:

ALIGN	محاذاة الصورة
ALT	النص البديل
ATTRIBUTES	تعيد قائمة بجميع واصفات هذا العنصر وقيمها
BORDER	عرض الإطار الخاص بالصورة
DISABLED	قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعّل وقيمتها الافتراضية هي FALSE
ID	مُعرّف وحيد لهذا العنصر
NAME	اسم الارتباط
ONSERVERCLICK	اسم التابع الذي سيتم تنفيذه عند النقر على هذه الوصلة
RUNAT	ويحدد أن عنصر التحكم من جهة المخدم Server يأخذ القيمة
SRC	محدد المصدر القياسي
STYLE	يعيد خصائص CSS التي سيتم تطبيقها على هذا العنصر
TYPE	نوع العنصر
VALUE	قيمة العنصر
VISIBLE	قيمة منطقية تحدد كون هذا العنصر مرئي أو لا

مثال:

```
<script  runat="server">
Sub button1(Source As Object, e As ImageClickEventArgs)
     p1.InnerHtml="You clicked the smiley button!"
End Sub
Sub button2(Source As Object, e As ImageClickEventArgs)
     p1.InnerHtml="You clicked the angry button!"
End Sub
</script>

<html>
<body>

<form runat="server">
<p>Click on one of the images:</p>
<p>
<input type="image" src="smiley.gif"
OnServerClick="button1" runat="server" width="32" height="32" />
</p>
<p>
```

```
OnServerClick="button2" runat="server" width="32" height="32" />
</p>
<p id="p1" runat="server" />
</form>

</body>
</html>
```

نلاحظ في هذا المثال استخدامنا النمط imageClickEventArgs عوضاً عن النمط EventArgs أثنا تعريف معالج حدث النقر على الصورة.

عنصر التحكم HTMLINPUTRADIOBUTTON

ATTRIBUTES	تعيد قائمة بجميع واصفات هذا العنصر وقيمها
CHECKED	**قيمة منطقية تحدد كون العنصر قد تم اختياره أم لا**
DISABLED	قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هيFALSE
ID	مُعرّف وحيد لهذا العنصر
NAME	**اسم الارتباط**
RUNAT	ويحدد أن عنصر التحكم من جهة المخدم يأخذ القيمةServer
STYLE	**يعيد خصائص CSSالتي سيتم تطبيقها على هذا العنصر**
TYPE	**نوع العنصر**
VALUE	**قيمة العنصر**
VISIBLE	**قيمة منطقية تحدد كون هذا العنصر مرئي أو لا**

مثال:

```
<script  runat="server">
Sub submit(Source As Object, e As EventArgs)
if r1.Checked=True then
   p1.InnerHtml="Your favorite color is red"
else
   if r2.Checked=True then
     p1.InnerHtml="Your favorite color is green"
   else
     if r3.Checked=True then
       p1.InnerHtml="Your favorite color is blue"
     end if
   end if
end if
End Sub
</script>

<html>
```

```
<form runat="server">
<p>Select your favorite color:
<br />
<input id="r1" name="col" type="radio" runat="server">Red</input>
<br />
<input id="r2" name="col" type="radio" runat="server">Green</input>
<br />
<input id="r3" name="col" type="radio" runat="server">Blue</input>
<br />
<input type="button" value="Submit" OnServerClick="submit" runat="server"/>
<p id="p1" runat="server" />
</form>

</body>
</html>
```

عنصر التحكم HTMLINPUTTEXT

ATTRIBUTES	تعيد قائمة بجميع واصفات هذا العنصر وقيمها
DISABLED	قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هيFALSE
ID	مُعرّف وحيد لهذا العنصر
MAXLENGHT	عدد المحارف الأقصى المسموح بإدخاله
NAME	اسم الارتباط
RUNAT	ويحدد أن عنصر التحكم من جهة المخدم Serverيأخذ القيمة
SIZE	عرض الحقل النصي
STYLE	التي سيتم تطبيقها على هذا CSSيحدد أو يعيد خصائص العنصر
TYPE	نوع العنصر
VALUE	قيمة العنصر
VISIBLE	قيمة منطقية تحدد كون هذا العنصر مرئي أو لا

عنصر التحكم HTMLSELECT

ATTRIBUTES	تعيد قائمة بجميع واصفات هذا العنصر وقيمها
DATAMEMBER	اسم جدول البيانات المرتبط مع عنصر التحكم
DATASOURCE	اسم مصدر البيانات المستخدم
DATATEXTFIELD	الحقل المطلوب إظهاره من حقول مصدر البيانات ضمن

العنصر	
الحقل المرتبط بالقيم المتعلقة بالعنصر من حقول مصدر البيانات	DATAVALUEFIELD
قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هي FALSE	DISABLED
مُعرّف وحيد لهذا العنصر	ID
يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو لايقوم بتحويل المحارف الخاصة إلى HTML	INNERHTML
يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو يقوم بتحويل المحارف الخاصة إلى HTML	INNERTEXT
قائمة بالعناصر ضمن القائمة المنسدلة	ITEMS
تحدد إمكانية اختيار أكثر من عنصر من القائمة في نفس الوقت	MULTIPLE
يجب تعيين هذه القيمة إلى اسم الإجرائية الفرعية التي تعالج حدث التغيير للعنصر المختار من القائمة	ONSERVERCHANGE
ويحدد أن عنصر التحكم من جهة Server يأخذ القيمة المخدم	RUNAT
الدليل وهو عدد صحيح يحدد العنصر الذي تم اختياره	SELECTEDINDEX
ارتفاع قائمة الاختيار يتم تعيينها إلى العدد 1 في حال الرغبة بإنشاء قائمة منسدلة.	SIZE
يعيد خصائص CSS التي سيتم تطبيقها على هذا العنصر	STYLE
قيمة العنصر المختار	VALUE
قيمة منطقية تحدد كون هذا العنصر مرئي أو لا	VISIBLE

مثال:

```
<script  runat="server">
Sub choose_image(Sender As Object, e As EventArgs)
  image1.Src = select1.Value
End Sub
</script>
```

```
<body>

<form runat="server">
<select id="select1" runat="server">
  <option value="smiley.gif">Smiley</option>
  <option value="angry.gif">Angry</option>
  <option value="stickman.gif">Stickman</option>
</select>
<input type="submit" runat="server" value="Display image"
OnServerClick="choose_image">
<br /><br />
<img id="image1" src="smiley.gif" runat="server" width="32" height="32" />
</form>

</body>
</html>
```

عنصر التحكم HTMLTABLE

ALIGN	تحدد محاذاة الجدول
ATTRIBUTES	تعيد قائمة بجميع واصفات هذا العنصر وقيمها
BGCOLOR	تحدد لون الخلفية
BORDER	تحدد سماكة الإطار للجدول
BORDERCOLOR	تحدد لون الإطار للجدول
CELLPADDING	تحدد المسافة بين الإطار ومحتوى الخلية
CELLSPACING	تحدد المسافة بين الخلايا.
DISABLED	قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هي FALSE
HEIGHT	تحدد ارتفاع الجدول
ID	مُعرّف وحيد لهذا العنصر
INNERHTML	يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو لايقوم بتحويل المحارف الخاصة إلى HTML
INNERTEXT	يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو يقوم بتحويل المحارف الخاصة إلى HTML
ROWS	يعيد مجموعة تحتوي جميع عناصر الصفوف الموجودة في الجدول
RUNAT	ويحدد أن عنصر التحكم من جهة Server يأخذ القيمة

المخدم	
التي سيتم تطبيقها على هذا العنصر CSSيعيد خصائص	**STYLE**
قيمة منطقية تحدد كون هذا العنصر مرئي أو لا	**VISIBLE**
تعيين عرض الجدول	**WIDTH**

مثال:

```
<script  runat="server">
Sub submit(sender As Object, e As EventArgs)
dim i,j
table1.BGColor="yellow"
table1.BorderColor="red"
for i=0 To table1.Rows.Count-1
  for j=0 To table1.Rows(i).Cells.Count-1
    table1.Rows(i).Cells(j).InnerHtml="Row " & i
  next
next
End Sub
</script>

<html>
<body>

<form runat="server">
<table id="table1" border="1" runat="server">
  <tr>
    <td>Cell 1</td>
    <td>Cell 2</td>
  </tr>
  <tr>
    <td>Cell 3</td>
    <td>Cell 4</td>
  </tr>
</table>
<br />
<input type="button" value="Change Contents" OnServerClick="submit" runat="server"/>
</form>

</body>
</html>
```

عنصر التحكم HTMLTABLECELL

المحاذاة للنص ضمن الخلية	ALIGN
تعيد قائمة بجميع واصفات هذا العنصر وقيمها	ATTRIBUTES
لون الخلفية	BGCOLOR

لون الإطار	BORDERCOLOR
عدد الأعمدة التي يجب أن تستمر عليها الخلية	COLSPAN
قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعّل وقيمتها الافتراضية هي FALSE	DISABLED
ارتفاع الخلية	HEIGHT
مُعرّف وحيد لهذا العنصر	ID
يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو لايقوم بتحويل المحارف الخاصة إلى HTML	INNERHTML
يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو يقوم بتحويل المحارف الخاصة إلى HTML	INNERTEXT
عدد الصفوف التي يجب أن تستمر عليها الخلية	ROWSPAN
ويحدد أن عنصر التحكم من جهة Server يأخذ القيمة المخدم	RUNAT
يعيد خصائص CSS التي سيتم تطبيقها على هذا العنصر	STYLE
المحاذاة الشاقولية للعناصر في الخلية	VALIGN
قيمة منطقية تحدد كون هذا العنصر مرئي أو لا	VISIBLE
عرض الخلية	WIDTH

مثال:

```
<script  runat="server">
Sub submit(sender As Object, e As EventArgs)
Dim row,numrows,numcells,j,i
row=0
numrows=rows1.Value
numcells=cells1.Value
for j=1 to numrows
Dim r As New HtmlTableRow()
row=row+1
for i=1 to numcells
Dim c As New HtmlTableCell()
c.Controls.Add(New LiteralControl("row " & j & ", cell " & i))
r.Cells.Add(c)
next
t1.Rows.Add(r)
t1.Visible=true
next
End Sub
</script>

<html>
<body>
```

```
<form runat="server">
<p>Table rows:
<select id="rows1" runat="server">
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
</select>
<br />Table cells:
<select id="cells1" runat="server">
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
</select>
<br /><br />
<input type="submit" value="Display Table" runat="server" OnServerClick="submit">
</p>
<table id="t1" border="1" runat="server" visible="false"/>
</form>

</body>
</html>
```

عنصر التحكم HTMLTABLEROW

محاذاة العناصر ضمن السطر	ALIGN
تعيد قائمة بجميع واصفات هذا العنصر وقيمها	ATTRIBUTES
لون الخلفية	BGCOLOR
لون الإطار	BORDERCOLOR
يعيد الخلايا الموجودة ضمن هذا السطر	CELLS
قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هي FALSE	DISABLED
ارتفاع السطر	HEIGHT
مُعرّف وحيد لهذا العنصر	ID
يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو لايقوم بتحويل المحارف الخاصة إلى HTML	INNERHTML
يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو يقوم بتحويل المحارف الخاصة إلى HTML	INNERTEXT
ويحدد أن عنصر التحكم من جهة المخدم يأخذ القيمة Server	RUNAT
يعيد خصائص CSS التي سيتم تطبيقها على هذا العنصر	STYLE
محاذاة العناصر شاقولياً ضمن السطر	VALIGN

قيمة منطقية تحدد كون هذا العنصر مرئي أو لا	VISIBLE

عنصر التحكم HTMLTEXTAREA

تعيد قائمة بجميع واصفات هذا العنصر وقيمها	ATTRIBUTES
عرض مربع النص	COLS
قيمة منطقية تحدد كون هذا العنصر مفعّل أم غير مُفَعَّل وقيمتها الافتراضية هي FALSE	DISABLED
مُعرّف وحيد لهذا العنصر	ID
يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو لايقوم بتحويل المحارف الخاصة إلى HTML	INNERHTML
يقوم بتعيين أو إعادة النص المحدد بتأشيرتي فتح وإغلاق هذا العنصر وهو يقوم بتحويل المحارف الخاصة إلى HTML	INNERTEXT
اسم مربع النص	NAME
اسم الإجرائية التي سيتم تنفيذها لمعالجة حدث التغيير على محتوى مربع النص	ONSERVERCHANGE
ويحدد أن عنصر التحكم من جهة Server يأخذ القيمة المخدم	RUNAT
التي سيتم تطبيقها على هذا العنصر CSS يعيد خصائص	STYLE
محتوى مربع النص	VALUE
قيمة منطقية تحدد كون هذا العنصر مرئي أو لا	VISIBLE

مثال:

```
<script  runat="server">
Sub submit(sender As Object, e As EventArgs)
  p1.InnerHtml = "<b>You wrote:</b> " & textarea1.Value
End Sub
</script>

<html>
<body>

<form runat="server">
Enter some text:<br />
<textarea id="textarea1" cols="35" rows="6" runat="server" />
<input type="submit" value="Submit" OnServerClick="submit" runat="server" />
<p id="p1" runat="server" />
</form>

</body>
</html>
```

الفصل الخامس

عنوان الموضوع:

عناصر تحكم نماذج الوب من جهة المخدم

الكلمات المفتاحية:

عنصر تحكم ، جهة المخدم ، وب ، تأشيرة، خصائص ، طرق، أحداث

ملخص:

قدمت ASP.NET تقنيات مميزة تحقق تحكم أكبر بمكونات صفحة الوب. وتعتبر عناصر التحكم من جهة المخدم هي أهم تلك التقنيات.

سنستعرض في هذه الجلسة مجموعة أخرى من عناصر التحكم من جهة المخدم نطلق عليها اسم عناصر تحكم نماذج الوب.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- عناصر التحكم من جهة المخدم
- خصائص عناصر التحكم من جهة المخدم
- تفاصيل وأمثلة متنوعة حول عناصر تحكم عناصر تحكم الوب

مقدمة

استعرضنا في الجلسة الماضية مجموعة من عناصر التحكم من جهة المخدم وأطلقنا عليها اسم عناصر تحكم HTML.

رأينا أن عناصر التحكم من جهة المخدم هي عناصر مُتَرجَمة ضمن الصفحة على المخدم، يمكن التفاعل معها، وتسمح بالانتقال إلى استخدام تقنية البرمجة المقادة بالاحداث مما يوفر بيئة تمكننا من كتابة نص برمجي أكثر وضوحاً وتسهل إزالة العلل منه.

لاحظنا أيضاً أنّ عناصر تحكم HTML لم تكن إلا مكافئات من جهة المخدم لتأشيرات HTML العادية والتي ما تزال تشكل آلية فعالة لإنشاء نماذج وصفحات تفاعلية .

سنستعرض في هذه الجلسة مجموعة أخرى من عناصر التحكم من جهة المخدم نطلق عليها اسم عناصر تحكم نماذج الوب.

عناصر تحكم نماذج وب

عناصر تحكم نماذج وب	المكافئةHTMLعناصر
<ASP:HyperLink>	<a>...
<ASP:LinkButton>	<a><img/>
<ASP:Image>	<img/>
<ASP:Panel>	...

<ASP:Label>	...
<ASP:Button>	<input type="submit"/> or
<ASP:TextBox>	____ أو أو <textarea>...</textarea>
<ASP:CheckBox>	☐
<ASP:RadioButton>	☐
<ASP:ImageButton>	<input type="image"/>
<ASP:Table>	<table>...</table>
<ASP:TableRow>	<tr>...</tr>
<ASP:TableCell>	<td>...</td>

o قد يبدو غريباً وجود مجموعة أخرى من عناصر تحكم نماذج الوب تقوم بعمل مشابه للعمل الذي تقوم به عناصر تحكم HTML. يعود السبب في ذلك إلى:

- تسهيل عملية بناء الأدوات والتطبيقات التي تولد تلقائياً واجهات المستخدم.
- تسهيل عملية إنشاء نماذج وب بأقل قدر من المعرفة بطريقة عمل عناصر تحكم HTML الأمر الذي يجعل النصوص البرمجية أقل عرضة لوجود أخطاء.

o بعكس عناصر تحكم HTML، تستخدم عناصر تحكم نماذج الوب أسماء موحدة لخصائصها التي لها نفس الدلالة. فعلى سبيل المثال، تمثل الخاصة SIZE لعنصر تحكم HTML المدعو listBox عدد الأسطر المرئية ضمن القائمة. بينما يصبح اسم هذه الخاصة Cols في عنصر تحكم HTML آخر كالعنصر TextArea. في حين تكون الخاصة WIDTH التي تعبر عن عرض عنصر تحكم، موحدة في جميع عناصر تحكم نماذج الوب التي نحتاج لتحديد عرضها.

o ليس هناك حاجة لأن يعرف المطور ما هو خرج HTML اللازم لضبط الأبعاد والخصائص بالشكل المطلوب، إذ تقتصر مهمته على تعيين قيم الخصائص المرتبطة بعناصر نماذج الوب فقط.

o تملك بعض عناصر نماذج الوب ميزات إضافية مساعدة مثل إضافة تلقائية للنص المرافق لمربع تحقق أو زر اختيار راديو.

خصائص وطرائق وأحداث عناصر تحكم نماذج الوب

تقدم عناصر تحكم نماذج الوب العديد من الخصائص، والطرائق، والأحداث. نـذكر منهـا تلـك المـشتركة الموروثة من الصف Webcontrols:

الخصائص، والطرق، والأحداث	الوصف
الخاصة Attributes	إعادة جميع الثنائيات (خاصة، قيمة الخاصة). يمكن استخدامها لقراءة الخصائص غير المعيارية (الخصائص التي لا تشكل جزءاً من التوصيف المعياري للغة HTML) أو للوصول إلى خصائص العنصر الخصوصية غير المُشارَكة.
الخاصة AccessKey	إعادة مفتاح الاختصار الذي يساعد على اختيار العنصر.
الخاصة BackColor	تحديد لون الخلفية.
الخاصة BorderColor	إعادة لون الإطار.
الخاصة BorderStyle	إعادة نمط إطار العنصر (منقط ، مستمر ....).
الخاصة BorderWidth	إعادة سماكة الإطار.
الخاصة ClientID	إعادة قيمة معرف العنصر المعينة من قبل ASP.NET
الخاصة Controls	إعادة عناصر التحكم (الأبناء) التابعة لعنصر التحكم الحالي (الأب).
الخاصة Enabled	إعادة ما إذا كان العنصر مفعلاً أم لا.
الخاصة EnableViewState	إعادة قيمة منطقية تحدد فيما إذا كان العنصر سيحافظ على قيمة المتغير Viewstate لهذا العنصر ولجميع أبنائه.
الخاصة Font	إعادة نوع الخط

الخاصة ForeColor	إعادة اللون المستخدم لمحتوى أو لنص عنصر التحكم
الخاصة Height	إعادة ارتفاع العنصر.
الخاصة ID	إعادة المعرف الخاص بعنصر التحكم هذا
الخاصة Page	إعادة مؤشر إلى كائن الصفحة الحاوي على عنصر التحكم
الخاصة Parent	يعيد مؤشر إلى أب عنصر التحكم وفق التسلسل الهرمي لعلاقة عناصر التحكم ضمن كائن الصفحة.
الخاصة Style	المطبقة على عنصر التحكمCSS يمثل مجموعة جميع الأنماط
الخاصة TabIndex	إعادة تسلسل عنصر التحكم ضمن ترتيب انتقال الاختيار لدى ضغط زر Tab
الخاصة ToolTip	إعادة النص الذي يظهر لدى تحريك مؤشر الفأرة فوق عنصر التحكم.
الخاصة Visible	إعادة قيمة منطقية تحدد إخفاء أو إظهار عنصر التحكم في خرج الصفحة.
الطريقة DataBind	تفعيل الربط بمصدر البيانات في عنصر التحكم وفي جميع أبنائه.
الخاصة Width	إعادة عرض عنصر التحكم.
الطريقة FindControl	يبحث ضمن الكائن الحالي عن عنصر تحكم معين
الطريقة HasControls	إعادة قيمة منطقية تحدد فيما إذا كان لعنصر التحكم عناصر تحكم ابناء.
الحدث DataBinding	يظهر هذا الحدث عند ربط عنصر تحكم بمصدر بيانات

خصائص ، طرق وأحداث عناصر تحكم نماذج الوب

خصائص وطرائق وأحداث عناصر تحكم نماذج الوب

تملك عناصر تحكم الوب طرائقها وخصائصها المميزة إضافةً إلى تلك التي ترثها. فيما يلي سرد لبعض أهـم تلك الطرائق والخصائص بحسب عناصر التحكم.

HyperLink	ImageUrl, NavigateUrl, Target, Text	-
LinkButton	CommandArgument, CommandName, Text, CausesValidation	OnClick, OnCommand
Image	AlternateText, ImageAlign, ImageUrl	-
Panel	BackImageUrl, HorizontalAlign, Wrap	-
Label	Text	
Button	CommandArgument, CommandName, Text, CausesValidation	OnClick, OnCommand
TextBox	AutoPostBack, Columns, MaxLength, ReadOnly, Rows, Text, TextMode, Wrap	OnTextChanged
CheckBox	AutoPostBack, Checked, Text, TextAlign	OnCheckChanged
RadioButton	AutoPostBack, Checked, GroupName, Text, TextAlign	OnCheckChanged
ImageButton	CommandArgument, CommandName, CausesValidation	OnClick, OnCommand
Table	BackImageUrl, CellPadding, CellSpacing, GridLines, HorizontalAlign, Rows	-
TableRow	Cells, HorizontalAlign, VerticalAlign	-
TableCell	ColumnSpan, HorizontalAlign, RowSpan, Text, VerticalAlign, Wrap	-
Literal	Text	-
PlaceHolder	-	-

تملك عناصر تحكم الوب طرائقها وخصائصها المميزة إضافةً إلى تلك التي ترثها. نجد في الجـدول الُرفـق سرد لبعض أهم تلك الطرائق والخصائص بحسب عناصر التحكم.

عناصر تحكم نماذج الوب

مثال 1:

نلاحظ في المثال التالي التشابه الواضح بين طريقة التعامل مع عناصر تحكم HTML وطريقة التعامل مـع عناصر تحكم نماذج الوب، فقد قمنا بإنشاء زر بمعرف button1 وقمنا بكتابة معالج الحـدث النـاتج عـن ضغطه، بشكل منفصل، في بداية المثال:

```
<script  runat="server">
Sub submit(Source As Object, e As EventArgs)
  button1.Text="You clicked me!"
End Sub
</script>

<html>
<body>

<form runat="server">
<asp:Button id="button1" Text="Click me!" runat="server" OnClick="submit"/>
</form>

</body>
</html>
```

مثال 2:

يتناول مثالنا الثاني عنصر تحكم الارتباط التشعبي، حيث نلاحظ أننا قمنا بتحديد صورة خاصـة بالارتبـاط بالإضافة إلى نص، وقمنا بتعيين المحدد المصدري القياسي (URL) الذي يعرف الإرتبـاط والنافـذة التـي سيجري فتح الارتباط فيها.

```
<html>
<body>

<form runat="server">
<asp:HyperLink ImageUrl="myImage.jpg" NavigateUrl="http://www.sk-www.com"
Text="Visit sk!" Target="_blank" runat="server" />
</form>
</body>
</html>
```

.

هناك تشابه واضح بين طريقة التعامل مع عناصر تحكم نماذج الوب وطريقة التعامل مع عناصر تحكم HTML. توضح الأمثلة المرافقة آلية وصيغة استخدام هذه العناصر.

خصائص عناصر تحكم نماذج الوب

الاختلاف في قيم الخصائص

يتميز استخدام عناصر تحكم HTML بالسهولة نظراً لأن قيم الخصائص تقتصر على سلاسل محارف كما هو الحال في تأشيرات HTML العادية.

يختلف الحال مع عناصر تحكم نماذج الوب إذ يمكن أن تكون هذه القيم عبارة عن أغراض أو عن قيم مُرقمة (Enumerated). فعلى سبيل المثال، نكتب عند محاذاة صورة إلى الجهة اليمنى لعنصر تحكم نموذج وب خاص بصورة:

```
<ASP:Image Src="mypic.gif" ImageAlign="Right" runat="server" />
```

يجري إعطاء قيمة للعنصر imageAlign.Right حيث يعرف العنصر عن طريق اسم الخاصة الترميز الذي يجب استخدامه.

أما إذا رغبنا بتعيين القيمة أثناء تنفيذ النص البرمجي، فيجب إضافة هذا السطر ضمن النص التنفيذي في الصفحة:

```
objMyImage.ImageAlign = ImageAlign.Right
```

أو إضافة النص:

```
objMyImage.ImageAlign = 2
```

يساعد الغرض TypeDescriptor على معرفة أسماء العناصر المُرقمة وذلك بعد استخدام import للربط مع الـ Namespace المناسب.

```
<%@Import Namespace="System.ComponentModel" %>
TypeDescriptor.GetConverter(GetType(HorizontalAlign)).ConvertFromSt
ring("Left")
```

أو نستطيع استخدام التحويل المباشر باستعمال :

```
Dim intValue = CType(HorizontalAlign.Left, Integer)
```

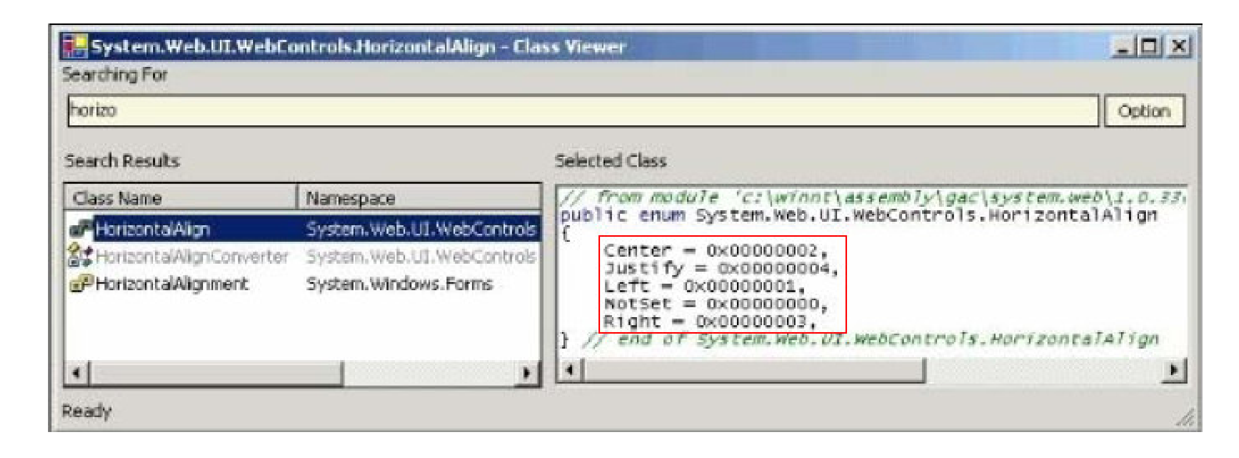
يتميز استخدام عناصر تحكم HTML بالسهولة نظراً لأن قيم الخصائص تقتصر على سلاسل محارف كما هو الحال في تأشيرات HTML العادية.

يختلف الحال مع عناصر تحكم نماذج الوب إذ يمكن أن تكون هذه القيم عبارة عن أغراض أو عن قيم مُرقمة

.(Enumerated)

خصائص عناصر تحكم نماذج الوب

يمكننا أيضاً ببساطة استخدام الأداة WinCV من إطار عمل NET. لتحديد القيم المُستَخدمة في ترقيم العناصر كما يظهر في الشكل:



حيث يكفي البحث بجزء من الاسم عن الخاصة ثم الضغط في الجزء الأيسر على الخاصة المطلوبة للحصول على القيم المتاحة لها.

هذا بالنسبة للخاصة المُرقمة ولكن ماذا بشأن الخاصة التي تشير إلى غرض.

لنأخذ حالة قد تبدو للوهلة الأولى بديهية، وهي حالة إسناد لون معين إلى أحد الخصائص BackColor, BorderColor) ForeColor,. قد لا نحتاج إذا كنّا نريد ضبط قيمة اللون إلى أكثر من إعطاء قيمة اللون مباشرةً:

```
<asp:textbox id="MyText" Text="This is a textbox" runat="server"
BackColor="Red" ForeColor="White" />
```

كما يمكننا تعريف غرض لون ومن ثم إسناده إلى خاصة اللون.

يمكننا أيضاً ببساطة استخدام الأداة WinCV من إطار عمل NET. لتحديد القيم المُستَخدمة في ترقيم العناصر

خصائص عناصر نماذج الوب

الصف System.Drwaing.Color

يعرف صف Color خصائص لجميع ألوان HTML القياسية (مثل AliceBlue، Antique، White) إضافة إلى توفير مجموعة من الطرائق لإنشاء غرض خاص باللون:

FromArgb	تقوم بإنشاء لون من خلال مركباته: الشفافية إضافةً إلى الأحمر والأخضر والأزرق.
FromKnownColor	تقوم بإنشاء كائن لون من اسم اللون وفقاً للتسميات المعتمدة في HTML
FromName	تقوم بإنشاء كائن لون من القيمة الستة عشرية للون بنفس الطريقة المتبعة فيHTML

مثال:

```
<%@Import Namespace="System.Drawing" %>
MyControl.BackColor = Color.FromName("ff0000")
```

يعرف صف Color خصائص لجميع ألوان HTML القياسية (مثل AliceBlue، Antique، White) إضافة إلى توفير مجموعة من الطرائق لإنشاء غرض خاص باللون:

خصائص عناصر تحكم نماذج الوب

صف Unit الخاص بواحدات القياس

يجري إسناد أغراض من الصف Unit إلى العديد من خصائص عناصر تحكم نماذج الوب، كحال خصائص Width، Height، Border، Width

فكما نلاحظ في هذا المثال قمنا باستخدام % و Px للدلالة على وحدة القياس :

```
<asp:image id="MyImage" Src="mypic.gif" runat="server"
Height="100px" Width="50%" />
```

إضافة إلى هذه الطريقة يمكننا استخدام الطرائق والخصائص التي يزودها الصف لإعطاء قيم أثناء عمل البرنامج.

يكون الصف Unit جزء من فضاء الأسماء الخاص بعناصر تحكم نماذج وب لذلك لا داعي لاستيراده لأنه

سيكون مستورد مسبقاً.

يمنح صف Unit الخصائص والطرق التالية:

الخاصيةType	تحدد نوع الوحدة المستخدمة يمكن أن تأخذ إحدى القيم Cm,Mm,Em,Ex,Inch,Percentage, Pica,Pixel,Point
الخاصيةValue	عدد الوحدات
الطريقةPercentage	تقوم بإنشاء كائن Unit من النمط Percentage
الطريقةPixel	تقوم بإنشاء كائن Unit من النمط Pixel
الطريقةPoint	تقوم بإنشاء كائن Unit من النمط Point

مثال:

```
MyControl.Height=Unit.Pixel(50)
```

في المثال، يمثل العدد 50 قيمة ارتفاع العنصر مقدرة بالبكسل.

يجري إسناد أغراض من الصف Unit إلى العديد من خصائص عناصر تحكم نماذج الوب، كحال خصائص Width، Height، Border، Width.

خصائص عناصر تحكم نماذج الوب

الخاصة AutoPostBack

عند بناء نموذج تفاعلي، يلزمنا في العديد من الحالات تفعيل عملية إرسال النموذج لدى اختيار عنصر من قائمة أو تفعيل مربع اختيار، بحيث يُسمح للمخدم بتحديث الصفحة عند الاستجابة لخيارات المستخدم.

كانت هذه العملية تتم سابقاً (في ASP) بواسطة حدث من جهة الزبون مرتبط بحدث اختيار أو تعديل حالة كحدث OnChange كما في النص:

```
<script language="javascript">
<!--
function __doPostBack(eventTarget, eventArgument) {
var theform = document.ctrl0;
theform.__EVENTTARGET.value = eventTarget;
theform.__EVENTARGUMENT.value = eventArgument;
theform.submit();
}
<input id="MyControl" type="checkbox" name="MyControl"
onclick="javascript:__doPostBack('MyControl','')" />
<input type="hidden" name="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" value="" />
// -->
```

أما في ASP.NET فيكفي إسناد القيمة True إلى الخاصة AutoPostBack حتى يجري إرسال الصفحة إلى المخدم بعد كل تعديل على عنصر التحكم الذي تم تعيين هذه الخاصة له.

الخاصية AutoPostBack:

عند بناء نموذج تفاعلي يلزمنا في العديد من الحالات تفعيل عملية إرسال للنموذج لدى الاختيار عنصر من قائمة مثلاً أو تفعيل مربع اختيار مما يسمح للمخدم بتحديث الصفحة لدى الاستجابة لخيارات المستخدم.
في ASP.NET يكفي تعيين قيمة الخاصية AutoPostBack إلى القيمة True فسيتم إرسال الصفحة إلى المخدم بعد كل تعديل على عنصر التحكم الذي تم تعيين قيمة هذه الخاصية له إلى True.

خصائص عناصر تحكم نماذج الوب

الخاصة EnableViewState لصفحة ASP.NET

يضطر عادةً المستخدم إلى إعادة ملء محتوى نموذج مجدداً بعد اختفاء عناصره نتيجة الخطأ في ملء أحد المعلومات. وقد كانت هذه الصفة مرافقة للعمل على نسخة ASP، إذ كانت قيم عناصر النموذج تختفي بعد إرساله وكان لا بد للمبرمج من كتابة نص برمجي لمعالجة هذه الحالة.

أما في ASP.NET فيكون الإبقاء على محتوى النموذج مُفَعَّل بصورة تلقائية ما لم يحدد المبرمج إلغاءه بصورة قسرية ضمن الصفحة بالعبارة:

```
<%@ Page EnableViewState="false" %>
```

يجب وضع هذه العبارة في بداية صفحة ASP.Net أو يمكن إيقاف هذه الخاصة لأحد العناصر فقط بتحديد EnableViewState="false" لهذا العنصر فقط. يمكن أن نستخدم هذه الحالة مثلاً عندما نتكلم عن حقل مخصص لإدخال كلمة سر.

مثال :

```
<script  runat="server">
Sub submit(sender As Object, e As EventArgs)
lbl1.Text="Hello " & txt1.Text & "!"
End Sub
</script>

<html>
<body>

<form runat="server">
Your name: <asp:TextBox id="txt1" runat="server"
EnableViewState="false"  />

<asp:Button OnClick="submit" Text="Submit" runat="server" />
<p><asp:Label id="lbl1" runat="server" /></p>
</form>

</body>
</html>
```

يضطر عادةً المستخدم إلى إعادة ملء محتوى نموذج مجدداً بعد اختفاء عناصره نتيجة الخطأ في ملء أحد المعلومات. وقد كانت هذه الصفة مرافقة للعمل على نسخة ASP، إذ كانت قيم عناصر النموذج تختفي بعد إرساله وكان لا بد للمبرمج من كتابة نص برمجي لمعالجة هذه الحالة.

أما في ASP.NET فيكون الإبقاء على محتوى النموذج مُفَعَّل بصورة تلقائية ما لم يحدد المبرمج إلغاءه بصورة قسرية ضمن الصفحة.

مثال

في المثال التالي استخدمنا عنصر تحكم نموذج وب checkBoxList حيث نلاحظ تفعيل الخاصة AutopostBack. سيقوم هذا البرنامج عند كل نقرة على أي عنصر من عناصر القائمة بإضافة اسم العنصر

إذا كان مُفعّلاً، على قيمة النص الخاص باللصاقة mess

```
<script  runat="server">
Sub Check(sender As Object, e As EventArgs)
   dim i
   mess.Text="<p>Selected Item(s):</p>"
   for i=0 to check1.Items.Count-1
     if check1.Items(i).Selected then
       mess.Text+=check1.Items(i).Text + "<br />"
     end if
   next
End Sub
</script>

<html>
<body>
<form runat="server">
<asp:CheckBoxList id="check1" AutoPostBack="True"
TextAlign="Right" OnSelectedIndexChanged="Check"
runat="server">
<asp:ListItem>Item 1</asp:ListItem>
<asp:ListItem>Item 2</asp:ListItem>
<asp:ListItem>Item 3</asp:ListItem>
<asp:ListItem>Item 4</asp:ListItem>
<asp:ListItem>Item 5</asp:ListItem>
<asp:ListItem>Item 6</asp:ListItem>
</asp:CheckBoxList>
<br />
<asp:label id="mess" runat="server"/>
</form>
</body>
</html>
```

نتيجة تنفيذ هذا المثال ستكون على الشكل

☑ Item 1
☐ Item 2
☑ Item 3
☑ Item 4
☑ Item 5
☐ Item 6

Selected Item(s):

Item 1
Item 3
Item 4
Item 5

مثال

في المثال التالي استخدمنا عنصر تحكم نموذج وب DropDownList .

سيقوم هذا البرنامج عند النقر على زر Submit بإظهار رسالة تحتوي على العنصر المختار.

```
<script  runat="server">
Sub submit(sender As Object, e As EventArgs)
   mess.Text="You selected " & drop1.SelectedItem.Text
End Sub
</script>

<html>
<body>

<form runat="server">
<asp:DropDownList id="drop1" runat="server">
<asp:ListItem>Item 1</asp:ListItem>
<asp:ListItem>Item 2</asp:ListItem>
<asp:ListItem>Item 3</asp:ListItem>
<asp:ListItem>Item 4</asp:ListItem>
<asp:ListItem>Item 5</asp:ListItem>
<asp:ListItem>Item 6</asp:ListItem>
</asp:DropDownList>
<asp:Button Text="Submit" OnClick="submit" runat="server"/>
<p><asp:label id="mess" runat="server"/></p>
</form>

</body>
</html>
```

ستكون نتيجة تنفيذ هذا المثال على الشكل:



You selected Item 3

مثال

نستخدم في المثال التالي عنصر تحكم نموذج وب DropDownList .

سيقوم هذا البرنامج عند النقر على زر Submit بإظهار رسالة تحتوي على العنصر المختار.

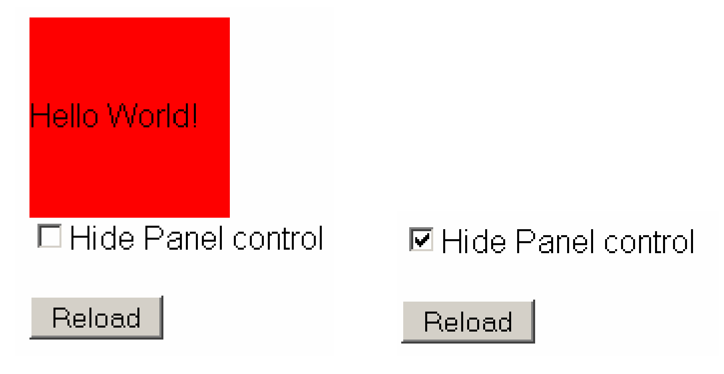
```
<script  runat="server">
Sub Page_Load(sender As Object, e As EventArgs)
   if check1.Checked then
     panel1.Visible=false
   else
     panel1.Visible=true
   end if
End Sub
</script>
```

```
<html>
<body>

<form runat="server">
<asp:Panel id="panel1"
runat="server" BackColor="#ff0000"
Height="100px" Width="100px">
ASP.NET
</asp:Panel>
<asp:CheckBox id="check1"
Text="Hide Panel control"
runat="server"/>
<br /><br />
<asp:Button Text="Reload" runat="server" />
</form>

</body>
</html>
```

ستكون نتيجة تنفيذ هذا المثال على الشكل:



مثال

نستخدم في المثال التالي عنصر تحكم نموذج وب table، tableCell,tableRow :

```
<html>
<body>

<form runat=server>
<asp:Table runat="server" CellPadding="5"
GridLines="horizontal" HorizontalAlign="Center">
   <asp:TableRow>
```

```
      <asp:TableCell>2</asp:TableCell>
    </asp:TableRow>
    <asp:TableRow>
      <asp:TableCell>3</asp:TableCell>
      <asp:TableCell>4</asp:TableCell>
    </asp:TableRow>
</asp:Table>
<br />
<asp:Table runat="server" CellPadding="5"
GridLines="vertical" HorizontalAlign="Center">
    <asp:TableRow>
      <asp:TableCell>1</asp:TableCell>
      <asp:TableCell>2</asp:TableCell>
    </asp:TableRow>
    <asp:TableRow>
      <asp:TableCell>3</asp:TableCell>
      <asp:TableCell>4</asp:TableCell>
    </asp:TableRow>
</asp:Table>
</form>

</body>
</html>
```

ستكون نتيجة تنفيذ هذا المثال على الشكل:

1	2
3	4

1	2
3	4

مثال

يوضح هذا المثال الخصائص المختلفة لعنصر تحكم النموذج من نوع TextBox

```
<html>
<body>
<form runat="server">
A basic TextBox:
<asp:TextBox id="tb1" runat="server" />
<br /><br />
A password TextBox:
<asp:TextBox id="tb2" TextMode="password" runat="server" />
<br /><br />
A TextBox with text:
<asp:TextBox id="tb3" Text="Hello World!" runat="server" />
<br /><br />
```

```
<asp:TextBox id="tb4" TextMode="multiline" runat="server" />
<br /><br />
A TextBox with height:
<asp:TextBox id="tb5" rows="5" TextMode="multiline" runat="server"
/>
<br /><br />
A TextBox with width:
<asp:TextBox id="tb6" columns="30" runat="server" />
</form>
</body>
</html>
```

ستكون نتيجة تنفيذ هذا المثال على الشكل:

A basic TextBox:

A password TextBox: ********

A TextBox with text: Hello World!

A multiline TextBox:

A TextBox with height:

A TextBox with width:

الفصل السادس

عنوان الموضوع:

عناصر تحكم التحقق من الإدخال.

الكلمات المفتاحية:

عنصر تحكم ، جهة المخدم ، وب ، تأشيرة، خصائص ، طرائق، أحداث

ملخص:

تشكل عملية التحقق من الإدخال جزء أساسي في أي تطبيق على الوب.

سنستعرض في هذه الجلسة مجموعة أخرى من عناصر التحكم من جهة المخدم نطلق عليها اسم عناصر تحكم التحقق من صحة الإدخال

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- عناصر تحكم التحقق من إدخال من جهة المخدم
- تفاصيل وأمثلة متنوعة حول عناصر تلك العناصر

عناصر تحكم السؤولة عن اعتمادية الدخل

Input Validation Control

لطالما كانت متطلبات التحقق من اعتمادية الدخل من أكثر المهام صعوبةً في بناء نماذج وب تفاعلية. إذ يمكن لهذه العملية أن تتم على مستوى المخدم أو على مستوى الزبون، ولكننا غالباً، ولغرض التوافقية مع عدة أنواع من المتصفحات نقوم بهذه العملية من طرف المخدم أو نحاول استخدام لغة JavaScript من جهة الزبون كونها أكثر توافقية مع أنواع المتصفحات المختلفة.

تقدم ASP.NET مجموعة من عناصر تحكم Input Validation تغطي أغلب السيناريوهات المحتملة لعمليات التحقق، إضافةً إلى تقديم عناصر تحكم قابلة للضبط يمكن استخدامها لإنشاء عناصر تحكم Input Validation غير قياسية.

فيما يلي قائمة بعناصر التحكم Input Validation مع توصيف كل منها:

الوصف	عنصر التحكم
يقوم بالتحقق من أن العنصر الذي يتم التأكد منه يحويعلى قيمة.	<asp:requiredFieldValidator>

يمكن استخدام هذا العنصر بالتعاون مع أخرى Input Validationعناصر لمعالجة العناصر الفارغة.	
يقوم بالتأكد من أن القيمة في العنصر الذي يتم التأكد منه، تقع ضمن المجال المحدد سواء كان هذا المجال يشمل مجموعة من المحارف أو الأرقام.	<asp:RangeValidator>
يقوم بالتحقق من أن القيمة التي يتم التأكد منها لعنصر التحكم مطابقة لقيمة عنصر تحكم آخر أو لقيمة محددة . إذا كان عنصر التحكم الذي يتم تقييمه خالي، لا تتم عملية التأكد.	<asp:CompareValidator>
يقوم بالتأكد من أن عنصر التحكم الذي يتم التأكد منه مطابق لتعبير نظامي. إذا كان عنصر التحكم الذي يتم تقييمه خالي، لا تتم عملية التأكد.	<asp:RegularExpressionValidator >
يسمح عنصر التحكم هذا بإنشاء عمليات تأكد مُعرَّفة من قبل المستخدم عبر تابع تحقق معين من طرف المخدم أو الزبون أو كلاهما. إذا كان عنصر التحكم الذي يتم تقييمه خالي، لا تتم عملية التأكد.	<asp:CustomValidator>
يقوم بإظهار تقرير بأخطاء عملية التأكد الجارية.	<asp:ValidationSummary>

عناصر تحكم السؤولة عن اعتمادية الدخل

Input Validation Control

لطالما كانت متطلبات التحقق من اعتمادية الدخل من أكثر المهام صعوبةً في بناء نماذج وب تفاعلية. إذ يمكن

لهذه العملية أن تتم على مستوى المخدم أو على مستوى الزبون، ولكننا غالباً، ولغرض التوافقية مع عدة أنواع من المتصفحات نقوم بهذه العملية من طرف المخدم أو نحاول استخدام لغة JavaScript من جهة الزبون كونها أكثر توافقية مع أنواع المتصفحات المختلفة.

تقدم ASP.NET مجموعة من عناصر تحكم Input Validation تغطي أغلب السيناريوهات المحتملة لعمليات التحقق، إضافةً إلى تقديم عناصر تحكم قابلة للضبط يمكن استخدامها لإنشاء عناصر تحكم Input Validation غير قياسية.

ما الذي تستطيع أن تفعله عناصر تحكم Input Validation

يجري ربط عنصر تحكم Input Validation أو أكثر، مع أحد عناصر تحكم الإدخال التي نود التأكد منها:

- عند إرسال الصفحة من قبل المستخدم يقوم كل عنصر تحكم Input Validation بالتأكد من القيمة الموجودة في عنصر التحكم المرتبط به ليرى فيما إذا كان سيتجاوز اختبار التحقق.
- في حال أي فشل في اختبار التحقق يقوم عنصر تحكم ValidationSummary بإظهار رسالة الخطأ المحددة حسب نتيجة الاختبار.
- تقوم عناصر تحكم Input Validation باكتشاف نوع المتصفح المستخدم وتوليد نص برمجي من جهة الزبون في الصفحة لإتمام عملية التحقق.
- يستخدم النص البرمجي من جهة الزبون لغة DHTML لإظهار محتوى عنصر التحكم على الصفحة بشكل ديناميكي.
- يقدم هذا الحل واجهة ذات استجابة عالية مشابهة بشكل كبير لما يمكن فعله باستخدام النصوص البرمجية المطورة يدوياً .

تؤمن عناصر تحكم Controls Input أيضاً الحماية من الاستخدام السيء لصفحات الموقع. فعملية التحقق من الدخل من طرف الزبون جيدة ولكنها غير آمنة، إذ يمكن للأشخاص الذين يرغبون في اساءة استخدام الموقع، إنشاء صفحاتهم الخاصة أو التعديل على الصفحات التي تصلهم بحيث لا يتم التحقق من المدخلات ويمكنهم بذلك ان ينتحلوا هوية مخدمك بإرسال معلومات غير صالحة.

على أي حال وبالرغم من كون عناصر تحكم Validation تقوم بالتحقق من الدخل من طرف الزبون، إلا أنها تقوم بنفس الفحص على المخدم عند إرسال الصفحة، وبهذا نستطيع الحصول على سرعة تجاوب في حالة

النصوص من جهة الزبون، وعلى الأمان في حالة النصوص البرمجية من جهة المخدم.

يمكننا إلغاء عملية التحقق من صلاحية الدخل من جهة الزبون في حال عدم الحاجة إليها، إضافة إلى إمكانية التحقق من صلاحية بيانات الدخل بصورة منفصلة، وتوليد رسالة خطأ مخصصة عوضاً عن استخدام عنصر التحكم ValidationSummary.

صف BaseValidator

- ترث جميع عناصر تحكم Validation خصائص وطرائق وأحداث الصف القاعدي BaseValidator.

- يُعتبر الصف BaseValidator جزءاً من فضاء الأسماء System.Web.UI.WebControls

- يقدم الصف BaseValidator مجموعة من الخصائص والطرائق المشتركة مع جميع عناصر تحكم Validation.

- أهم تلك الخصائص والطرائق:

الأحداث	الخصائص	عنصر التحكم
-	InitialValue	RequiredFieldValidator
OnServerValidate	MaximumValue, MinimumValue, Type	RangeValidator
-	DisplayMode, ShowHeaderText, ShowMessageBox, ShowSummary	validationSummary

- ترث جميع عناصر تحكم Validation خصائص وطرائق وأحداث الصف القاعدي

.BaseValidator

- يُعتبر الصف BaseValidator جزءاً من فضاء الأسماء System.Web.UI.WebControls

- يقدم الصف BaseValidator مجموعة من الخصائص والطرائق المشتركة مع جميع عناصر تحكم Validation.

عنصر التحكم من نوع RequieredFieldValidator

يتطلب هذا العنصر إدخال قيمة ما للعنصر لينجح اختبار التحقق.

مثال :

```
A Required Value:
<input type="text" id="txtRequired" size="20" runat="server" />
<asp:RequiredFieldValidator id="valRequired" runat="server"
ControlToValidate="txtRequired"
ErrorMessage="* You must enter a value in the first textbox"
Display="dynamic">
*
</asp:RequiredFieldValidator>
```

نلاحظ بأننا قمنا بتحديد عنصر التحكم txtRequired ليتم التحقق منه بواسطة عنصر التحكم RequieredFieldValidator وذلك بإسناد القيمة txtRequiered إلى الخاصة ControlToValidate.

قمنا أيضاً بتحدبد رسالة الخطأ التي ستتم إعادتها إلى المستخدم من خلال الخاصة ErrorMessage.

تحدد الخاصة Display فيما إذا كان عنصر التحكم سيحتل مكان في الصفحة حتى لو لم يتم إظهاره.

يتطلب هذا العنصر إدخال قيمة ما للعنصر لينجح اختبار التحقق.

عنصر التحكم CompareValidator

يستخدم عنصر التحكم هذا لمقارنة قيمة مدخلة مع قيمة عنصر تحكم آخر أو قيمة ثابتة.

يعتبر المثال التالي استمرار للمثال السابق حيث نلاحظ أننا أضفنا حقل إدخال نصي جديد وقمنا باسناد القيمة txtCompare إلى الخاصة ControlToValidate والقيمة txtRequired إلى الخاصة ControlToCompare حيث تحدد هاتان الخاصتان اسماء عناصر التحكم التي يراد التحقق من تطابق مدخلاتها.

تحدد الخاصة Operator نوع المقارنة بين قيمتي عنصري التحكم المقارنين ويمكن أن تأخذ إحدى القيم: (Equal,greaterThan,LessThanOrEqual,..)

```
The Same Value Again:
<input type="text" id="txtCompare" size="20" runat="server" />
<asp:CompareValidator id="valCompare" runat="server"
ControlToValidate="txtCompare"
ControlToCompare="txtRequired"
Operator="Equal"
ErrorMessage="* You must enter same value in the second textbox"
Display="dynamic">
*
</asp:CompareValidator>
```

كما يمكن استخدام عنصر التحكم CompareValidator لمقارنة دخل عنصر تحكم مع قيمة ثابتة كما في المثال التالي:

```
A Date after 27rd July 2005:
<input type="text" id="txtCompareDate" size="10" runat="server" />
<asp:CompareValidator id="valCompareDate" runat="server"
ControlToValidate="txtCompareDate"
ValueToCompare="27/6/2005"
Operator="GreaterThan"
Type="Date"
ErrorMessage="* The Date must be later than 27rd July 2005"
Display="dynamic">
*
```

```
</asp:CompareValidator>
```

نلاحظ هنا أننا استخدمنا عنصر التحكم CompareValidator لمقارنة القيمة المدخلة في العنصر txtCompareDate مع القيمة الثابتة "27/6/2005". نلاحظ أننا لم نجر المقارنة اعتماداً على النمط التلقائي وهو String بل حددنا نمط البيانات على أنه تاريخ باستخدام التعبير Type=”Date”.

تكون الأنماط المتاحة للبيانات هي Currency,Double,Date,Integer,String.

عنصر التحكم CompareValidator:

يستخدم عنصر التحكم هذا لمقارنة قيمة مدخلة مع قيمة عنصر تحكم آخر أو قيمة ثابتة.

يعتبر المثال التالي استمرار للمثال السابق حيث نلاحظ أننا أضفنا حقل إدخال نصي جديد وقمنا باسناد القيمة txtCompare إلى الخاصة ControlToValidate والقيمة txtRequired إلى الخاصة ControlToCompare حيث تحدد هاتان الخاصتان اسماء عناصر التحكم التي يراد التحقق من تطابق مدخلاتها.

تحدد الخاصة Operator نوع المقارنة بين قيمتي عنصري التحكم المقارنين ويمكن أن تأخذ إحدى القيم: (Equal,greaterThan,LessThanOrEqual,..)

كما يمكن استخدام عنصر التحكم CompareValidator لمقارنة دخل عنصر تحكم مع قيمة ثابتة.

عنصر التحكمRange Validator

يستخدم عنصر التحكم هذا لتحديد المجال المسموح للقيم التي يمكن أن يأخذها عنصر تحكم ما.

لا بد في حالة عنصر التحقق هذا، من تحديد ثلاث خصائص أساسية: تتعلق الأولى بنمط البيانات المستخدمة، أما الثانية والثالثة فتحدد القيم العليا والدنيا المسموح إدخالها من النمط المحدد.

تستخدم الخاصة الأولى التعبير Type وتستخدم الخاصة الثانية التعبير MaximumValue، وتستخدم الخاصة الثالثة MinimumValue.

ففي النص البرمجي التالي نلاحظ أننا نقوم بإسناد القيمة Integer للخاصة Type واسناد القيم 1 و 10 للخاصتين MaximumValue و MinimumValue

```
A Number between 1 and 10:
<input type="text" id="txtRange" size="5" runat="server" />
<asp:RangeValidator id="valRange" runat="server"
ControlToValidate="txtRange"
MaximumValue="10"
MinimumValue="1"
Type="Integer"
ErrorMessage="* The Number must between 1 and 10"
Display="dynamic">
*
</asp:RangeValidator>
```

يستخدم عنصر التحكم هذا لتحديد المجال المسموح للقيم التي يمكن أن يأخذها عنصر تحكم ما.

لا بد في حالة عنصر التحقق هذا، من تحديد ثلاث خصائص أساسية: تتعلق الأولى بنمط البيانات المستخدمة، أما الثانية والثالثة فتحدد القيم العليا والدنيا المسموح إدخالها من النمط المحدد.

تستخدم الخاصة الأولى التعبير Type وتستخدم الخاصة الثانية التعبير MaximumValue، وتستخدم الخاصة الثالثة MinimumValue.

عنصر تحكم التحقق باستخدام التعابير النظامية

تُعرَّف التعابير النظامية بأنها عبارة عن مجموعة من الرموز والإشارات التي تستخدم لتحديد تنسيق معين للبيانات. من أشهر هذه الرموز:

. ^ + * [ ] \ ? |

تمتلك ASP.NET عنصر تحكم Validation خاص بالتعابير النظامية يقوم بالتأكد من مطابقة قيمة ما مدخلة مع تعبير نظامي محدد. يتم تحديد صيغة التعبير النظامي ضمن الخاصة ValidationExpression.

سنستخدم في المثال التالي عنصر التحكم الخاص بالتعابير النظامية للتأكد من أن التعبير المدخل ضمن عنصر

تحكم النص txtRegExpr هو تعبير عنوان بريد إلكتروني صحيح.
نلاحظ أننا استخدمنا التعبير النظامي *..*.@*. للخاصة ValidationExpression حيث يعبر رمز .
عن أي محرف و * عن تكرار لأي محرف

```
Match Expression "<b>.*@.*\..*</b>":
<input type="text" id="txtRegExpr" size="40" runat="server" />
<asp:RegularExpressionValidator id="valRegExpr" runat="server"
ControlToValidate="txtRegExpr"
ValidationExpression=".*@.*\..*"
ErrorMessage="* Your entry does not match the regular expression"
Display="dynamic">
*
</asp:RegularExpressionValidator>
```

تُعرَّف التعابير النظامية بأنها عبارة عن مجموعة من الرموز والإشارات التي تستخدم لتحديد تنسيق معين للبيانات.

تمتلك ASP.NET عنصر تحكم Validation خاص بالتعابير النظامية يقوم بالتأكد من مطابقة قيمة ما مدخلة مع تعبير نظامي محدد. يتم تحديد صيغة التعبير النظامي ضمن الخاصة ValidationExpression.

عنصر تحكم التحقق المخصص

عندما تكون عملية التحقق التي نود إجراءها أعقد من أن يلبيها أي عنصر من عناصر التحكم التي استعرضناها مسبقاً ، نستطيع استخدام عنصر تحكم التحقق المخصص CustomValidator.

نستخدم في المثال التالي عنصر التحكم Comparevalidator للتأكد من إدخال قيمة أكبر من 100 وعنصر التحكم CustomValidator لفحص القيمة وتقييمها:

```
A Prime Number over 100:
<input type="text" id="txtCustom" size="5" runat="server" />
<asp:CompareValidator id="valComparePrime" runat="server"
ControlToValidate="txtCustom"
ValueToCompare="100"
Operator="GreaterThan"
```

```
Type="Integer"
ErrorMessage="* The Prime Number must be greater than 100"
Display="dynamic">
*
</asp:CompareValidator>
<asp:CustomValidator id="valCustom" runat="server"
ControlToValidate="txtCustom"
ClientValidationFunction="ClientValidate"
OnServerValidate="ServerValidate"
ErrorMessage="* Your knowledge of prime numbers is not optimal"
Display="dynamic">
*
</asp:CustomValidator>
```

يعتمد عنصر تحكم CustomValidator خاصتي ClientValidationFunction و OnserverValidate لتحديد اسماء التوابع المخصصة التي سنقوم بإنشائها للتحقق من القيمة.

عندما تكون عملية التحقق التي نود إجراءها أعقد من أن يلبيها أي عنصر من عناصر التحكم التي استعرضناها مسبقاً ، نستطيع استخدام عنصر تحكم التحقق المخصص CustomValidator.

يعتمد عنصر تحكم CustomValidator خاصتي ClientValidationFunction و OnserverValidate لتحديد اسماء التوابع المخصصة التي سنقوم بإنشائها للتحقق من القيمة.

التابع الخاص بعنصر التحكم المخصص:

يسمح عنصر التحكم المخصص باستخدام تابع مخصص من جهة المخدم ومن جهة الزبون .
نستخدم عادةً JavaScript ذفي كتابة النص البرمجي للتابع من جهة الزبون لضمان أعلى توافقية مع أنواع المتصفحات وذلك لضرورة إدراج هذا النص البرمجي ضمن الصفحة.

```
ControlToValidate="txtCustom"
ValueToCompare="100"
Operator="GreaterThan"
Type="Integer"
ErrorMessage="* The Prime Number must be greater than 100"
Display="dynamic">
*
</asp:CompareValidator>
<asp:CustomValidator id="valCustom" runat="server"
ControlToValidate="txtCustom"
ClientValidationFunction="ClientValidate"
OnServerValidate="ServerValidate"
ErrorMessage="* Your knowledge of prime numbers is not optimal"
Display="dynamic">
```

```
*
</asp:CustomValidator>
<script language="JavaScript">
<!--
// client-side validation function for CustomValidator
function ClientValidate(objSource, objArgs) {
var blnValid = true;
var intNumber = objArgs.Value;
if (intNumber % 2 == 1) {
var intDivisor = Math.floor(intNumber / 3);
if (intDivisor > 2) {
for (var i = 3; i <= intDivisor; i = i + 2) {
if (intNumber % intDivisor == 0) {
blnValid = false;
break;
}
}
}
else
blnValid = false;
}
else
blnValid = false;
objArgs.IsValid = blnValid;
return;
}
//-->
</script>
```

نلاحظ كيف توفر عناصر تحكم التحقق مرجع إلى نفسها هو ObjSource، بالإضافة إلى غرض بإسم ObjArgs يتضمن عناصر هذا التابع.

نحصل على قيمة عنصر التحكم من خلال الحقل Value لـــ ObjArgs ونسند قيمة إلى الحقل IsValid الخاص بـــ ObjArgs لتحديد فيما إذا نجح التحقق أم لا.

أما فيما يخص التحقق من طرف المخدم ، نستخدم نفس المبدأ وهنا يجب علينا استخدام VB.NET لإنشاء التابع من جهة المخدم.

في هذه الحالة يقوم التابع على المخدم باستلام مؤشر إلى عنصر التحكم والغرض ServerValidateEventArgs الحاوي على قيمة عنصر التحكم كقيمة للخاصة Value:

```
blnValid = false;
```

```
}
else
blnValid = false;
objArgs.IsValid = blnValid;
return;
}
//-->
</script>
Sub ServerValidate(objSource As Object, objArgs As
ServerValidateEventArgs)
Dim blnValid As Boolean = True
Try
Dim intNumber As Integer = objArgs.Value
'check that it's an odd number
If intNumber Mod 2 = 1 Then
'get the largest possible divisor
Dim intDivisor As Integer = intNumber \ 3
If intDivisor > 2 Then
Dim intLoop As Integer
'check using each divisor in turn
For intLoop = 3 To intDivisor Step 2
If intNumber Mod intDivisor = 0 Then
blnValid = False
Exit For
End If
Next
Else
blnValid = False
End If
Else
blnValid = False
End If
Catch objError As Exception
blnValid =False
Finally
objArgs.IsValid = blnValid
End Try
End Sub
```

يسمح عنصر التحكم المخصص باستخدام تابع مخصص من جهة المخدم ومن جهة الزبون .

نستخدم عادةً JavaScript ذفي كتابة النص البرمجي للتابع من جهة الزبون لضمان أعلى توافقية مع أنواع المتصفحات وذلك لضرورة إدراج هذا النص البرمجي ضمن الصفحة.

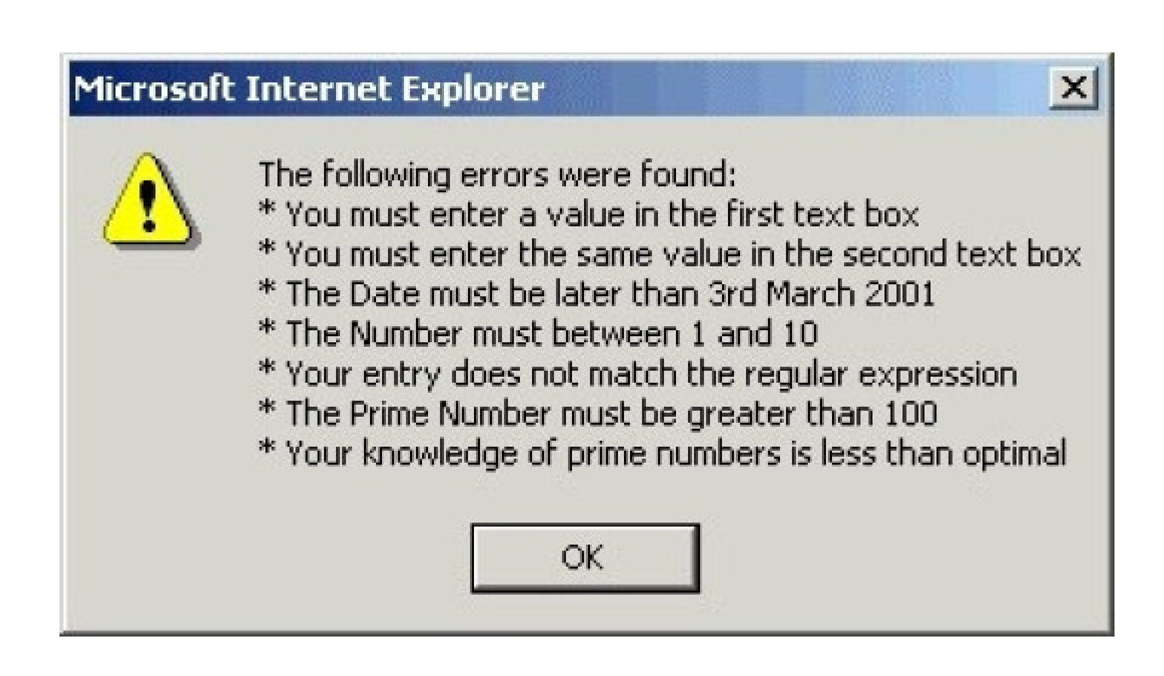
عنصر تحكم ValidationSummary:

يقوم هذا العنصر بإظهار قائمة بالأخطاء عند إرسال الصفحة.

سنقوم في المثال التالي بتوضيح كيفية عمل عنصر التحكم هذا.
قمنا في البداية بتحديد ترويسة ملخص الأخطاء الذي سوف يظهره عنصر التحكم وذلك باسناد قيمة للخاصة HeaderText، ثم قمنا بإسناد القيمة True إلى الخاصة ShowSummary بحيث يتم إظهار قيمة الخاصة ErrorMessage المرتبطة بكل عنصر تحكم (يجري التحقق منه) في حال فشل الاختبار على هذا العنصر.

```
<asp:ValidationSummary id="valSummary" runat="server"
HeaderText="<b>The following errors were found:</b>"
ShowSummary="True" DisplayMode="List" />
```

تمكن الخاصة DisplayMode من اختيار طريقة إظهار رسائل الخطأبحيث يمكن أن نسند إلى هذه الخاصة القيم (List,BulletList,SingleParagraph. يمكننا أيضاً استخدام الخاصة ForColor لضبط لون الرسالة .
تعتبر الخاصة ShowMessageBox أحد الخصائص المفيدة أيضاً إذ تحدد هذه الخاصة إظهار ملخص نتائج الخطأ ضمن علبة حوار عوضاً عن الصفحة نفسها.



يقوم هذا العنصر بإظهار قائمة بالأخطاء عند إرسال الصفحة.

تعطيل وتفعيل النص البرمجي من جهة الزبون أو المخدم والخاصة EnableClientScript

يمكن تعطيل أي نص البرمجي خاص بعناصر تحكم، على المخدم أو الزبون، بتعديل قيمة الخاصةُ Enabled و EnabledClientScript

```
Sub Page_Load()
Dim objValidator As BaseValidator
For Each objValidator In Page.Validators
objValidator.Enabled = lstEnabled.SelectedItem.Text
objValidator.EnableClientScript = lstClientScript.SelectedItem.Text
Next
End Sub
```

الفصل السابع والثامن

عنوان الموضوع:

ربط البيانات

الكلمات المفتاحية:

عنصر تحكم، قيمة، قائمة، صف، عمود، متكررة، وحيدة، غرض، حقل، ربط، بيانات، نص برمجي، خاصة، قاعدة بيانات.

ملخص:

نستعرض في هذا الفصل اسلوب ربط عناصر التحكم.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- مبدأ ربط البيانات
- كيف نستطيع ربط عناصر تحكم إلى قيمة وحيدة
- كيف يمكننا ربط عناصر التحكم إلى مجموعة من القيم
- كيف نغير مظهر طريقة إظهار عناصر التحكم المرتبطة

عناصر تحكم القوائم والربط مع مصادر البيانات

تحتاج أغلب مواقع وتطبيقات الوب إلى الوصول إلى مصادر البيانات في مرحلة ما. وقد استطاعت ASP تأمين هذا الاتصال مع العديد من أنواع مصادر البيانات مثل قواعد البيانات العلائقية، ومخدمات البريد الالكتروني، ووثائق XML، والملفات النصية، كما استطاعت تأمين العمليات عليها وتنسيقها وإظهارها بعدة طرق. ولكن العمل مع ASP كان يتطلب كتابة نصوص برمجية لتأدية هذه الأعمال مما يجعلها تستهلك الوقت وتسبب مصاعب في الإدارة وفي إزالة العلل.

قدمت ASP.NET طرق جديدة لإدارة وإظهار البيانات من كل الأنواع. إذ تجري إدارة البيانات في ASP.NET عبر مجموعة متكاملة من عناصر التحكم من جهة المخدم التي تم تصميمها للعمل مع جميع أنواع قواعد البيانات وليس مع البيانات العلائقية ولا مع وثائق XML فقط.
الجديد هنا أننا لسنا مضطرون لمعرفة مصدر البيانات بشكلٍ كامل، ولا معرفة كيفية استخراجها حتى نستطيع استخدام عناصر التحكم الخاصة بالبيانات.
التصميم غير المتصل وغير الآني (Off Line) المعتمد في إطار عمل NET. يعني أنه يمكن لنا فصل قواعد العمل التي تقوم بالتعامل مع البيانات عن تلك التي تتعامل مع إظهارها.

لذا سنتعرف في هذه الجلسة على:

- مبدأ ربط البيانات؛
- كيف نستطيع ربط عناصر تحكم إلى قيمة وحيدة
- كيف يمكننا ربط عناصر التحكم إلى مجموعة من القيم
- كيف نغير مظهر طريقة إظهار عناصر التحكم المرتبطة

ربط البيانات

يصعب تعريف مبدأ ربط البيانات بصورة عامة في لغات البرمجة مثل Visual basic وفي التطبيقات مثل Microsoft Access.

يُعرَّف مبدأ ربط البيانات بأنه الطريقة التي ترتبط فيها عناصر تحكم على نموذج، بقيم من حاويات بيانات مثل مجموعة من سجلات قواعد المعطيات. حيث يستخدم النموذج للتحرك بين السجلات ويتم إظهار كل قيمة من قيم الحقول بصورة آلية ضمن عناصر التحكم.

ترتبط عناصر التحكم بالحقول في مجموعة السجلات ولسنا بحاجة لكتابة أي نص برمجي لإظهار القيم أو تحديث مصدر البيانات الأصلي.

عموماً، لاتسمح طبيعة البروتوكول HTTP باستخدام الطريقة التقليدية (زبون–مخدم) المستخدمة في Visual Basic و Access. لذا تضمن متصفح IE4 بعض مكونات COM من طرف الزبون والتي سمحت باستخدام تقنيات مماثلة لتلك المستخدمة في VB و Access مع البروتوكول HTTP. تمت تسمية هذه الكونات بمكونات ربط البيانات من جهة الزبون، ولكن العديد من الشكوك التي ظهرت والتي تعلقت بأمان المعلومات أدت إلى عدم الاعتماد على هذه الطريقة بشكل أساسي.

إدارة جميع الأعمال على المخدم:

إن التعدد والاختلاف بين التجهيزات وتطبيقات المستعرض من طرف الزبون جعل من استخدام تقنية تتعلق بالمتصفح أمراً غير محبب، لذا سعت مايكروسوفت في NET. إلى تقديم دعم لأغلب أنواع الزبائن إما ببناء تطبيقات تتحسس نوع التطبيق الزبون وتغير تصرفها بشكل يتوافق معه، أو بوضع جميع الفعاليات على المخدم.

في الكثير من الحالات تُعد عملية إدارة جميع الأعمال على المخدم خطة جيدة لأنها تسمح لنا بإنشاء عناصر تحكم ضمن الخرج، كما تسمح بزيادة أمان التطبيقات.
تم الاتجاه في بيئة NET. إلى مبدأ ربط البيانات من جهة المخدم والاستفادة من ميزاته في توفير الوقت والتقليل من كتابة النصوص البرمجية والعمل بتقنيات مشابهة لتلك المستخدمة في VB و Access ولكن عبر البروتوكول HTTP في بيئة غير متصلة كبيئة الوب.

إظهار البيانات

استطعنا في نسخة ASP3.0 والنسخ السابقة لها استخدام مكونات أو تقنيات مثل ADO لإنشاء غرض recordSet يحتوي الصفوف والبيانات التي نود إظهارها.
وكنا نضطر –لوضعها على الصفحة– إلى الدخول في حلقة تكرارية عبر الصفوف لاستخراج القيم، وتنسيقها ووضعها ضمن الخرج كما يظهر في المثال التالي:

```
...' assuming we've got a Recordset object containing the data ...
Response.Write "<table>"
Response.Write "<tr><th>Date</th>"
Response.Write "<th>Subject</th>"
Response.Write "<th>User Name</th>"
Response.Write "<th>Content</th></tr>"
Do While Not objRecs.EOF
strDate = FormatDateTime(objRecs("dtDate").value, vbLongDate)
Response.Write "<tr><td>" & strDate & "</td>"
Response.Write "<td>" & objRecs("tSubject").value & "</td>"
Response.Write "<td>" & objRecs("tUserName").value & "</td>"
Response.Write "<td>" & objRecs("tContent").value & "</td></tr>"
objRecs.MoveNext
Loop
Response.Write "</table>"
objRecs.Close
```

لأداء نفس العمل في ASP.NET يكفي كتابة النص البرمجي التالي:

```
<!-- the server control located in the HTML section of the page -->
<ASP:DataGrid id="MyDataGrid" runat="server" />
...' assuming we've got a DataView object containing the data ...
MyDataGrid.DataSource = objDataView
MyDataGrid.DataBind()
```

يقوم عنصر تحكم المخدم بأداء نفس العمل الذي يقوم به النص البرمجي ASP 3.0 آلياً: إذ يقوم بإنشاء جدول HTML بأسماء الحقول في الصف الأول، متبوعاً بسلسلة من الصفوف التي تحتوي القيم الخاصة بكل صف في مصدر البيانات، ونستطيع بكل بساطة تغيير تنسيق الجدول بتعديل خصائص عنصر التحكم ASP:DataGrid.

فعلى سبيل المثال نستطيع إضافة ترتيب آلي أو تقسيم إلى صفحات أو إظهار رقم الصف فقط بإسناد قيم لبعض الخصائص وإضافة سطر أو سطرين إلى النص البرمجي.

تقدم ASP.NET عناصر تحكم يمكن استخدامها لتأمين إظهار البيانات سواء كانت متكررة كما في مثالنا السابق، أو كانت تتعامل مع عنصر بيانات وحيد. يجري كل ذلك من خلال ربط البيانات من جهة المخدم.

إظهار البيانات

استطعنا في نسخة ASP3.0 والنسخ السابقة لها استخدام مكونات أو تقنيات مثل ADO لإنشاء غرض recordSet يحتوي الصفوف والبيانات التي نود إظهارها.

وكنا نضطر –لوضعها على الصفحة– إلى الدخول في حلقة تكرارية عبر الصفوف لاستخراج القيم، وتنسيقها ووضعها ضمن الخرج.

يقوم عنصر تحكم المخدم بأداء نفس العمل الذي يقوم به النص البرمجي ASP 3.0 آلياً: إذ يقوم بإنشاء جدول HTML بأسماء الحقول في الصف الأول، متبوعاً بسلسلة من الصفوف التي تحتوي القيم الخاصة بكل صف في مصدر البيانات، ونستطيع بكل بساطة تغيير تنسيق الجدول بتعديل خصائص عنصر التحكم ASP:DataGrid. فعلى سبيل المثال نستطيع إضافة ترتيب آلي أو تقسيم إلى صفحات أو إظهار رقم الصف فقط بإسناد قيم لبعض الخصائص وإضافة سطر أو سطرين إلى النص البرمجي.

تقدم ASP.NET عناصر تحكم يمكن استخدامها لتأمين إظهار البيانات سواء كانت متكررة كما في مثالنا السابق، أو كانت تتعامل مع عنصر بيانات وحيد. يجري كل ذلك من خلال ربط البيانات من جهة المخدم.

صيغة ربط البيانات

يعتمد مبدأ ربط البيانات من جهة المخدم على جعل ASP.NET تدرج قيمة أو عدة قيم من مصدر البيانات

على الصفحة أو ضمن عنصر تحكم في الصفحة.

تستخدم الصيغة الأساسية بنية مشابهة لتلك المستخدمة في كتلة النص البرمجي من جهة المخدم باستخدام إشارة # كدلالة على أن الصيغة هي صيغة ربط بيانات.

```
<%# name-of-data-source %>
```

لا يمكننا وضع النص المراد تنفيذه ضمن هذه الكتلة مع أنها تبدو ككتلة نص برمجي من جهة المخدم، ولكنها ليست كذلك، ولا يمكننا سوى استخدام التعبيرات الخاصة بربط البيانات ضمن هذه الكتلة.

نميز حالتين أساسيتين في ربط البيانات:

الحالة الأولى: ربط بيانات قيمة وحيدة:
تُستخدم هذه الحالة عندما يكون لدينا قيمة وحيدة نود ربطها مع عنصر تحكم كحال رغبتنا بإسناد قيمة إلى خاصة ما من خصائص عنصر التحكم. في هذه الحالة تُستخدم القيمة المرتبطة ليتم إسنادها إلى الخاصة المعبرة عن القيمة التي سيظهرها عنصر التحكم (مثلاً خصائص TEXT و VALUE). يُعتبر هذا الأسلوب مناسباً لعناصر التحكم التي تظهر لها قيمة وحيدة مثل <INPUT>، ASP:TextBox، ASP:HyperLink .

الحالة الثانية: ربط بيانات قيم متكررة:
تُستخدم هذه الطريقة عندما يحتوي مصدر البيانات على أكثر من قيمة. فعلى سبيل المثال يمكننا ربط قائمة أو مجموعة أو مجموعة صفوف مع عنصر تحكم يمكنه إظهار أكثر من قيمة.
تتضمن هذه الحالة عناصر تحكم مثل <SELECT>، ASP:ListBox، ASP:CheckBoxList.

على كل حال تُستخدم نفس التقنيات الأساسية في تطبيق أي نوع من نوعي الربط.

ربط بيانات قيمة وحيدة

عندما نقوم بربط عنصر تحكم إلى قيمة وحيدة مثل أحد الخصائص، أو إحدى الطرائق أو أحد التعبيرات، يمكننا استخدام أحد الصيغ البسيطة التالية:

```
<%# property-name %>
```

أو

```
<%# method-name (parameter1, parameter2,…) %>
```

أو

```
<%# expression%>
```

يمكننا أن نرى من الصيغ السابقة أن هناك عدة مصادر محتملة للقيمة التي يمكننا ربطها إلى عنصر التحكم:

- قيمة الخاصة المُصرَّح عنها في الصفحة أو في عنصر تحكم أو في غرض آخر.
- القيمة المُعادة من طريقة والمُصرَّح عنها في الصفحة أو في عنصر تحكم أو في غرض آخر.
- نتيجة تعبير معين.

```
ReadOnly Property ImageURL() As String
Get
'read-only property for the Page
Dim strURL As String
'some code would be here to calculate the value
'we just set it to a fixed value for illustration
strURL = "myimage.gif"
Return strURL
End Get
End Property
```

هناك عدة مصادر محتملة للقيمة التي يمكننا ربطها إلى عنصر التحكم:

- قيمة الخاصة المُصرَّح عنها في الصفحة أو في عنصر تحكم أو في غرض آخر.
- القيمة المُعادة من طريقة والمُصرَّح عنها في الصفحة أو في عنصر تحكم أو في غرض آخر.
- نتيجة تعبير معين.

استخدام عناصر التحكم مع القيم المرتبطة

تكمن الميزة الحقيقية في استخدام القيم المرتبطة في كون كتلة ربط البيانات قابلة للاستخدام مع عناصر تحكم أخرى ويمكن الحصول على قيمة أحد العناصر عن طريق عنصر تحكم آخر.

فعلى سبيل المثال يمكن ربط عنصر تحكم LABEL إلى عنصر تحكم TextBox بإسناد الخاصة TEXT لعنصر التحكم TextBox كقيمة للخاصة TEXT في عنصر تحكم LABEL.

```
<form runat="server">
<ASP:TextBox id="MyTextBox" runat="server" />
```

```
<ASP:Label id="MyLabel" Text="<%# MyTextBox.Text %>" runat="server"
/>
<input type="submit" />
</form>
<script language="VB" runat="server">
Sub Page_Load()
DataBind()
End Sub
</script>
```

في كل مرة يتم فيها إرسال الصفحة، ستكون القيمة في عنصر التحكم LABEL مأخوذة من عنصر التحكم TextBox المسمى myTextBox ولإتمام هذا العمل، لايلزمنا سوى كتابة النص البرمجي الخاص باستدعاء الطريقة DataBind.

لاتنحصر هذه التقنية بإسناد قيمة الخاصة لعنصر تحكم، إذ يمكننا أيضاً استخدامها لإدراج قيمة في أي عنصر تحكم. فعلى سبيل المثال، لإعطاء قيمة للخاصة Src في عنصر تحكم ASP:Image نستخدم الصيغة:

```
<ASP:Image Src="<%# ImageURL %>" ImageAlign="middle" runat="server"
/>
```

لتحديد النص ومحدد المصدر القياسي URL لعنصر تحكم ASP:HyperLink يمكننا استخدام:

```
<ASP:Hyperlink Text="<%# ImageURL %>" NavigateUrl="<%# ImageURL %>"
runat="server" />
```

تعمل هذه التقنية بشكل مشابه مع عناصر تحكم HTML وحتى مع تأشيرات HTML العادية.

```
<input type="text" value="<%# ImageURL %>" />
```

```
<input type="submit" value="<%# ImageURL %>" />
```

```
<a href="<%# ImageURL %>"><%# ImageURL %></a>
```

```
<a href="http://mysite.com/images/<%# ImageURL %>">
View the file named '<%# ImageURL %>'</a>
```

استخدام عناصر التحكم مع القيم المرتبطة

تكمن الميزة الحقيقية في استخدام القيم المرتبطة في كون كتلة ربط البيانات قابلة للاستخدام مع عناصر تحكم

أخرى ويمكن الحصول على قيمة أحد العناصر عن طريق عنصر تحكم آخر.

فعلى سبيل المثال يمكن ربط عنصر تحكم LABEL إلى عنصر تحكم TextBox بإسناد الخاصة TEXT لعنصر التحكم TextBox كقيمة للخاصة TEXT في عنصر تحكم LABEL.

لاتنحصر هذه التقنية بإسناد قيمة الخاصة لعنصر تحكم، إذ يمكننا أيضاً استخدامها لإدراج قيمة في أي عنصر تحكم.

تفعيل الربط

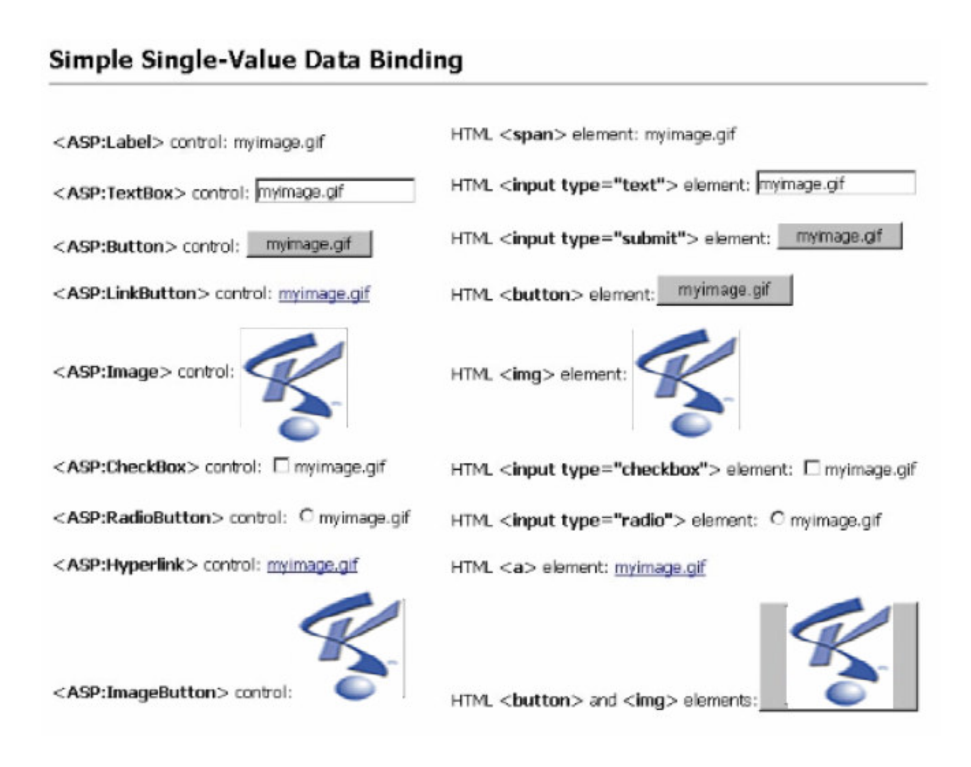
بعد تحديد كتلة ربط البيانات في الصفحة، يتوجب علينا تفعيل الربط وذلك عند انطلاق حدث ما (غالباً مايكون حدث تحميل الصفحة) لإدراج القيم المناسبة.
إذ يجري إهمال كتلة الربط من قبل محرك ASP أثناء ترجمة الصفحة، إذا لم يجري التفعيل، ولا يجري عندها استبدال كتل الربط بالقيم المطلوبة.

تجري عملية تفعيل الربط باستخدام طريقة DataBind المتوفرة في أكثر من سوية ضمن هرمية الصفحة.
- لربط جميع عناصر التحكم ضمن الصفحة نستخدم الطريقة DataBind الخاصة بالغرض Page.
هي الطريقة الوحيدة الفعالة في حال استخدام كتلة الربط مع عناصر غير مصممة لربط البيانات (مثل تأشيرات HTML).

- لربط عنصر تحكم وحيد فقط نستخدم الطريقة DataBind لهذا العنصر فقط وتستخدم فقط في عناصر التحكم المصممة خصيصاً للتعامل مع ربط البيانات.
- لربط صف أوعنصر من غرض من مصدر بيانات في عنصر تحكم نستخدم الطريقة DataBind لهذا الغرض. وهذه الطريقة كسابقتها لا تطبق إلا على العناصر المخصصة للتعامل مع ربط البيانات.

مثال:

يوضح المثال كيفية استخدام ربط البيانات لقيمة وحيدة باستخدام عدة أنواع من عناصر التحكم.




```
@%>Page Language="VB<%"

!>DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN<"
>html><head<
>title>Simple Single-Value Data Binding</title<
>style type="text/css<"
body, td {font-family:Tahoma,Arial,sans-serif; font-size:10pt{
input {font-family:Tahoma,Arial,sans-serif; font-size:9pt{
.heading {font-family:Tahoma,Arial,sans-serif; font-size:14pt; font-weight:bold{
.subhead {font-family:Tahoma,Arial,sans-serif; font-size:12pt; font-weight:bold; padding-
bottom:5px{
.cite {font-family:Tahoma,Arial,sans-serif; font-size:8pt{
/>style></head<
>body bgcolor="#ffffff<"
>span class="heading">Simple Single-Value Data Binding</span><hr</
<---------------------------------------------------------------------------!>

>form runat="server<"

>table border="0<"
>tr><td nowrap="nowrap<"


>b>&lt;ASP:Label&gt;</b> control:
>ASP:Label Text="<%# ImageURL %>" runat="server" /><p</

>b>&lt;ASP:TextBox&gt;</b> control:
>ASP:TextBox Text="<%# ImageURL %>" runat="server" /><p</

>b>&lt;ASP:Button&gt;</b> control:
```

```
>ASP:Button Text="<%# ImageURL %>" runat="server" /><p</

>b>&lt;ASP:LinkButton&gt;</b> control:
>ASP:LinkButton Text="<%# ImageURL %>" runat="server" /><p</

>b>&lt;ASP:Image&gt;</b> control:
>ASP:Image Src="<%# ImageURL %>" ImageAlign="middle" runat="server" /><p</

>b>&lt;ASP:CheckBox&gt;</b> control:
>ASP:CheckBox Text="<%# ImageURL %>" runat="server" /><p</

>b>&lt;ASP:RadioButton&gt;</b> control:
>ASP:RadioButton Text="<%# ImageURL %>" runat="server" /><p</

>b>&lt;ASP:Hyperlink&gt;</b> control:
>ASP:Hyperlink Text="<%# ImageURL %>" NavigateUrl="<%# ImageURL %>" runat="server"
/><p</

>b>&lt;ASP:ImageButton&gt;</b> control:
>ASP:ImageButton ImageUrl="<%# ImageURL %>" runat="server</ "


/>td><td nowrap="nowrap">     </td><td nowrap="nowrap<"


HTML <b>&lt;span&gt;</b> element:
>span><%# ImageURL %></span><p</

HTML <b>&lt;input type="text"&gt;</b> element:
>input type="text" value="<%# ImageURL %>" /><p</

HTML <b>&lt;input type="submit"&gt;</b> element:
>input type="submit" value="<%# ImageURL %>" /><p</

HTML <b>&lt;button&gt;</b> element:
>button><%# ImageURL %></button><p</

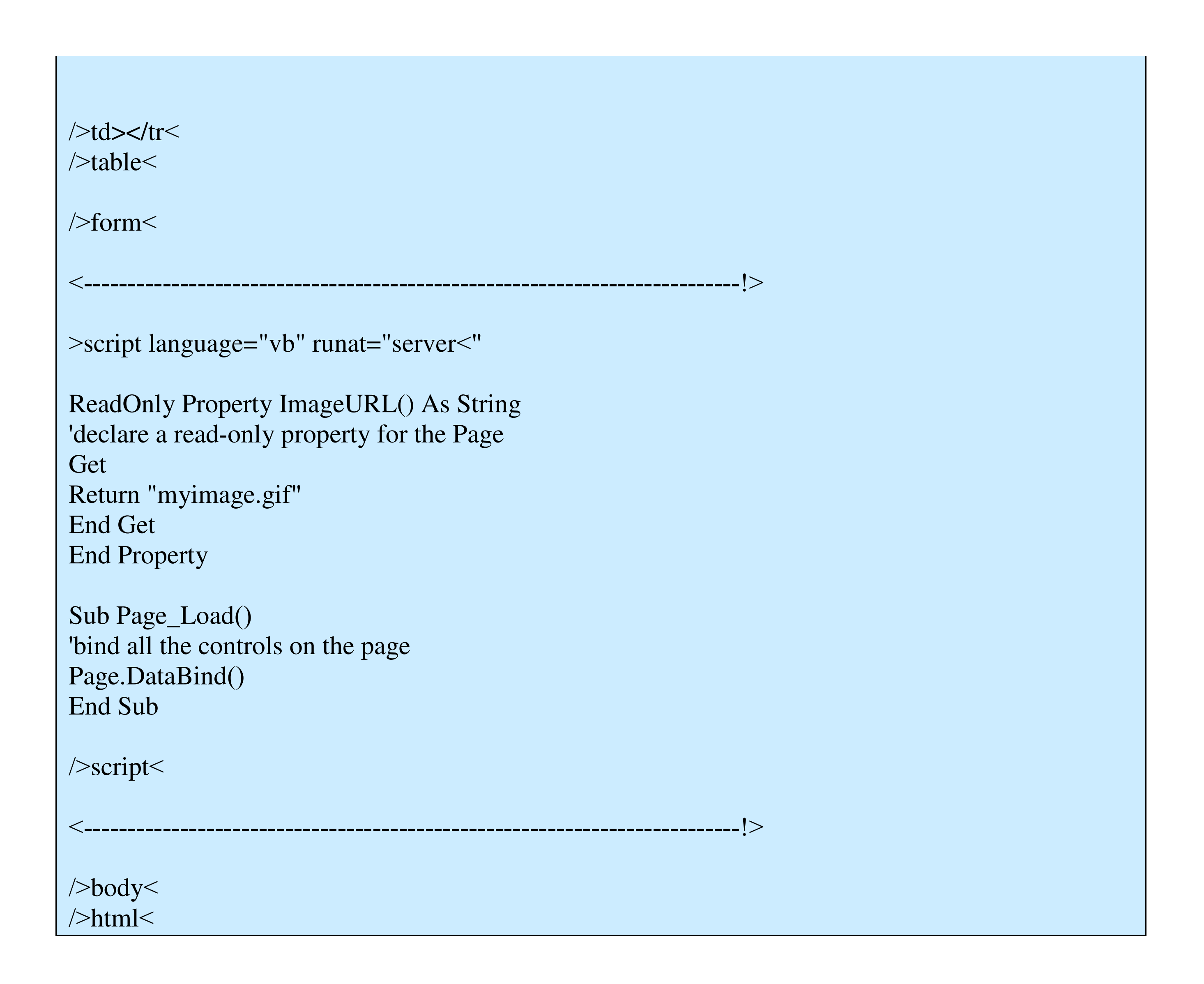
HTML <b>&lt;img&gt;</b> element:
>img src="<%# ImageURL %>" align="middle" /><p</

HTML <b>&lt;input type="checkbox"&gt;</b> element:
>input type="checkbox" /><%# ImageURL %><p</

HTML <b>&lt;input type="radio"&gt;</b> element:
>input type="radio" /><%# ImageURL %><p</

HTML <b>&lt;a&gt;</b> element:
>a href="<%# ImageURL %>"><%# ImageURL %></a><p</

HTML <b>&lt;button&gt;</b> and <b>&lt;img&gt;</b> elements:
>button><img src="<%# ImageURL %>"></button<
```



```
/>td></tr<
/>table<

/>form<

<-------------------------------------------------------------------------!>

>script language="vb" runat="server<"

ReadOnly Property ImageURL() As String
'declare a read-only property for the Page
Get
Return "myimage.gif"
End Get
End Property

Sub Page_Load()
'bind all the controls on the page
Page.DataBind()
End Sub

/>script<

<-------------------------------------------------------------------------!>

/>body<
/>html<
```

بعد تحديد كتلة ربط البيانات في الصفحة، يتوجب علينا تفعيل الربط وذلك عند انطلاق حدث ما (غالباً مايكون حدث تحميل الصفحة) لإدراج القيم المناسبة.

إذ يجري إهمال كتلة الربط من قبل محرك ASP أثناء ترجمة الصفحة، إذا لم يجري التفعيل، ولا يجري عندها استبدال كتل الربط بالقيم المطلوبة.

تجري عملية تفعيل الربط باستخدام طريقة DataBind المتوفرة في أكثر من سوية ضمن هرمية الصفحة.

- لربط جميع عناصر التحكم ضمن الصفحة نستخدم الطريقة DataBind الخاصة بالغرض Page.

هي الطريقة الوحيدة الفعالة في حال استخدام كتلة الربط مع عناصر غير مصممة لربط البيانات (مثل

تأشيرات HTML).

- لربط عنصر تحكم وحيد فقط نستخدم الطريقة DataBind لهذا العنصر فقط وتستخدم فقط في عناصر التحكم المصممة خصيصاً للتعامل مع ربط البيانات.
- لربط صف أوعنصر من غرض من مصدر بيانات في عنصر تحكم نستخدم الطريقة DataBind لهذا الغرض. وهذه الطريقة كسابقتها لا تطبق إلا على العناصر المخصصة للتعامل مع ربط البيانات.

ربط البيانات ذات القيم المتكررة

تُعتبر عملية ربط البيانات بقيمة وحيدة تقنية مفيدة. ولكن ربط البيانات كتقنية يصبح أكثر أهمية عند وجود صفوف متكررة من القيم التي نريد إظهارها، مثل مجموعة من الصفوف من قاعدة بيانات علائقية أو من مصفوفة قيم.

توفر ASP.NET ثماني عناصر تحكم قوائم جرى تصميمها للعمل مع القيم المتكررة حيث تملك مجموعة من الخصائص والطرق التي تسمح لها بالاتصال بمصادر البيانات. ثم تقوم بصورة آلية بإنشاء عنصر إظهار أو صف إظهار لكل صف ضمن مصدر البيانات.

كمثال على ذلك نأخذ عنصر التحكم من قائمة <SELECT> الذي يقوم بإنشاء العدد المناسب من عناصر <OPTION> لإظهار جميع الصفوف ضمن مصدر البيانات.

تتلخص أحد الميزات الواضحة في هذه التقنية في عدم اضطرارنا إلى كتابة أي نص HTML بنفسنا لإظهار هذا التصميم، إذ يجري إنشاؤه بشكل آلي من قبل عنصر التحكم (مع احتفاظنا بالطبع بالقدرة على إضافة عناصر HTML لتخصيص الخرج حسب الحاجة).

عناصر تحكم القوائم المصممة للتعامل مع الربط المتكرر:

نستعرض فيما يلي العناصر الثمانية المصممة لاستخدام ربط البيانات من جهة المخدم:

- عنصر التحكم <SELECT> وهو يستخدم الصف system.web.ui.htmlControls.htmlSelect
- عنصر التحكم ASP:ListBox الذي يستخدم الصف system.web.ui.WebControls.ListBox
- عنصر التحكم ASP:DropDown الذي يستخدم الصف system.web.ui.WebControls.DropDown
- عنصر التحكم ASP:CheckBoxList الذي يستخدم الصف system.web.ui.webControls.CheckBoxList

- عنصر التحكم ASP:RadioButtonList الذي يستخدم الصف system.web.ui.webControls.RadioButtonList
- عنصر التحكم ASP:Repeter الذي يستخدم الصف system.web.ui.webControls.Repeter
- عنصر التحكم ASP:DataList الذي يستخدم الصف system.web.ui.webControls.DataList
- عنصر التحكم ASP:DataGrid الذي يستخدم الصف system.web.ui.webControls.DataGrid

تُعتبر عملية ربط البيانات بقيمة وحيدة تقنية مفيدة. ولكن ربط البيانات كتقنية يصبح أكثر أهمية عند وجود صفوف متكررة من القيم التي نريد إظهارها، مثل مجموعة من الصفوف من قاعدة بيانات علائقية أو من مصفوفة قيم.

توفر ASP.NET ثماني عناصر تحكم قوائم جرى تصميمها للعمل مع القيم المتكررة حيث تملك مجموعة من الخصائص والطرق التي تسمح لها بالاتصال بمصادر البيانات. ثم تقوم بصورة آلية بإنشاء عنصر إظهار أو صف إظهار لكل صف ضمن مصدر البيانات.

كمثال على ذلك نأخذ عنصر التحكم من قائمة <SELECT> الذي يقوم بإنشاء العدد المناسب من عناصر <OPTION> لإظهار جميع الصفوف ضمن مصدر البيانات.

تتلخص أحد الميزات الواضحة في هذه التقنية في عدم اضطرارنا إلى كتابة أي نص HTML بنفسنا لإظهار هذا التصميم، إذ يجري إنشاؤه بشكل آلي من قبل عنصر التحكم (مع احتفاظنا بالطبع بالقدرة على إضافة عناصر HTML لتخصيص الخرج حسب الحاجة).

عناصر تحكم القوائم المصممة للتعامل مع الربط المتكرر:

نستعرض فيما يلي العناصر الثمانية المصممة لاستخدام ربط البيانات من جهة المخدم:

- عنصر التحكم <SELECT> وهو يستخدم الصف system.web.ui.htmlControls.htmlSelect
- عنصر التحكم ASP:ListBox الذي يستخدم الصف system.web.ui.WebControls.ListBox
- عنصر التحكم ASP:DropDown الذي يستخدم الصف system.web.ui.WebControls.DropDown

- عنصر التحكم ASP:CheckBoxList الذي يستخدم الصف system.web.ui.webControls.CheckBoxList
- عنصر التحكم ASP:RadioButtonList الذي يستخدم الصف system.web.ui.webControls.RadioButtonList
- عنصر التحكم ASP:Repeter الذي يستخدم الصف system.web.ui.webControls.Repeter
- عنصر التحكم ASP:DataList الذي يستخدم الصف system.web.ui.webControls.DataList
- عنصر التحكم ASP:DataGrid الذي يستخدم الصف system.web.ui.webControls.DataGrid

خصائص عناصر التحكم المرتبطة المتكررة

جرى تصميم عناصر التحكم لاستخدامها من جهة المخدم بحيث تقدم مجموعة من الخصائص والطرق لإدارة عملية الربط. نستعرض فيما يلي لائحة بهذه الخصائص:

الخاصة	وصف
DataTextField	تحدد حقل أو عمود مصدر البيانات الذي يحتوي على القيم التي ستُستخدم لإظهارها. على سبيل المثال القيم المستخدمة ضمن عنصر التحكم <OPTION>كنص للعنصر ListBox.
DataValueField	تحدد حقل أو عمود مصدر البيانات الذي يحتوي القيم التي ستُستخدم من أجل الخاصة VALUEفي العناصر التابعة لعنصر التحكم. على سبيل المثال القيم المستخدمة في الخاصة VALUE لعناصر <OPTION> التابعة لعنصر التحكم ListBox.
DataTextFormatString	تحدد تنسيق سلسلة المحارف المستخدمة مع القيم المحددة من العمود أو الحقل المُشار إليه بالخاصة DataTextList عند إظهار هذه القيم ضمن العنصر. على سبيل المثال نستخدم '{0:C}' أو لتنسيق العملة '{0:dddd MMMM dd,yyyy}' لتنسيق التاريخ.
DataMember	تحدد مجموعة الصفوف التي سيتم الربط بها عند احتواء مصدر البيانات على أكثر من مجموعة صفوف. على سبيل المثال اسم الجدول عند الربط مع غرض DataSet.

طرق وأحداث عناصر التحكم المرتبطة المكررة

تقدم جميع عناصر التحكم المُصممة للتعامل مع القيم المكررة، طريقتين على الأقل للاستخدام مع البيانات المرتبطة:

الطريقة	الوصف
DataBind	بملء عناصر التحكم بالبيانات من تقوم مصادر البيانات أي باختصار تفعيل الربط الذي قمنا بضبطه أثناء التصريح عن عنصر التحكم.
FindControl	يُستخدم لاستعادة مؤشر إلى عنصر التحكم الابن ضمن عنصر التحكم المرتبط.

الأحداث الخاصة بعناصر التحكم المرتبطة المكررة:

هناك مجال واسع للأحداث التي يتم إطلاقها بواسطة عنصر تحكم قائمة، يتعلق بعضها بنوع عنصر التحكم .على كل حال هناك حدثان مشتركان بين أنواع عناصر تحكم القائمة المصممة لربط البيانات:

الحدث	الوصف
DataBinding	يظهر هذا الحدث من أجل كل صف في مصدر البيانات. يتم تمرير محتوى الصف إلى الحدث ضمن معاملات الحدث حيث يمكن فحص وتعديل بيانات الصف أثناء ملء عنصر التحكم بالمعلومات.
SelectedindexChange	يظهر هذا الحدث عندما يتم إرسال الصفحة مع العناصر المختارة إلى المخدم، وهو يسمح للنص البرمجي بإجراء التغييرات على الإظهار لتعكس خيارات المستخدم.

مصادر البيانات لربط القيم المتكررة

بعد أن اطّلعنا على عناصر التحكم سنستعرض فيما يلي مصادر البيانات التي يمكن ربطها بهذا النوع من العناصر.

يمكن من الناحية الفنية، ربط عناصر تحكم القوائم إلى أي نوع من مصادر البيانات التي تستخدم IEnumerable، ICollection أو IListSource . هذا يعني من الناحية العملية أنّ عناصر تحكم القوائم يمكن ربطها إلى:

- Collection مثل مجموعة القيم الخاصة بـــ Request.Form أو مجموعة الجداول في غرض Dataset أو أي مجموعة نُنشأها بأنفسنا
- ArrayList: وهي تحتوي مجموعة بسيطة من القيم. وهي طريقة جيدة عموماً إذا كنا نريد إظهار خيارات في ListBox مثلاً
- HashTable: تحتوي عناصر يمكن الوصول إليها عبر مفتاح كحال الغرض Dictionary. يكون لكل عنصر مفتاح وقيمة مما يجعل مصادر البيانات هذه مثالية في حالة عناصر تحكم القوائم حيث يختلف النص المراد إظهاره عن القيمة المراد إعادتها عند اختيار عنصر
- غرض DataView: يحتوي هذا الغرض على صفوف من أغراض DataTable يجري تأهيلها من قاعدة البيانات أو يجري إنشاؤها وتأهيلها يدوياً عن طريق النص البرمجي
- غرض DataReader: الذي يوفر اتصال مخصص للقراءة وباتجاه واحد مع قاعدة البيانات. يمكن أن يحوي صف أو مجموعة صفوف ويتم التعامل معه بصورة مشابهة لما يحصل مع غرض Dataview . نستخدم عادةً الغرض DataReader لأغراض الأداء حيث يكون أداء الغرض DataReader أسرع في أغلب الحالات

صيغة استخدام ربط البيانات للقيم المتكررة

عند ربط عنصر التحكم بقيمة وحيدة كحال الربط مع خاصة أو الربط مع القيمة المعادة من طريقة بالصيغة:

```
<%# name-of-data-source %>
```

تحتوي مصادر البيانات المرتبطة بقيم متكررة على أكثر من حقل. ففي حال كان مصدر البيانات من النوع HashTable الذي يحتوي على مفتاح وقيمة أو كان من النوع DataReader، يجب أن نكون قادرين على تحديد العمود أو الحقل المراد ربط عنصر التحكم إليه.

تكون كثير من عناصر تحكم القوائم قادرة على إظهار أكثر من قيمة لكل عنصر في القائمة. فعلى سبيل المثال، نستطيع في العنصر <SELECT> استخدام قيمة لمحتوى العنصر <OPTION> وقيمة للخاصة VALUE لهذا العنصر.

```
<Select>
<Option Value="Value1"> Text1</Option>
<Option Value="Value2"> Text2</Option>
</Select>
```

إسقاط الحقول من مصدر البيانات على خصائص عنصر التحكم:

هناك طريقتان لإسقاط أو وصل حقل محدد من كل صف في مصدر البيانات إلى خصائص عنصر تحكم.
يعتمد اختيار الطريقة على عنصر التحكم الذي نقوم بربطه:

- إذا كان عنصر التحكم يدعم القوالب يمكننا التصريح عن قالب لكل صف سيتم إظهاره من قبل عنصر التحكم.
- إذا كان عنصر التحكم لا يدعم القوالب نستطيع إسناد الحقول في مصدر البيانات إلى خصائص عنصر التحكم بضبط الخصائص أثناء عمل البرنامج.

تكون عناصر التحكم في ASP.NET التي تدعم القوالب هي: Repeater، DataList و DataGrid وعليه، يمكننا لأي من هذه العناصر التصريح عن قالب ووضع تعليمات ربط البيانات ضمنه.

```
Key:<%# Container.DataItem.Key%>
```

أو

```
Value:<%Container.DataItem.Value%>
```

عند الربط مع أغراض Collection أو DataView أو DataReader نحدد الخاصة، الحقل أو العمود نفسه:

```
Value from DataView/DataReader: <%# Container.DataItem (“bookName”) %>
```

أو

```
Value from Collection: <%# Container.DataIem(“forColor”) %>
```

مثلاً نستطيع استخدام عنصر تحكم Repeater لإظهار قيم Key و Value ضمن غرض HashTable:

```
<ASP:Repeater id="MyRepeater" runat="server">
<ItemTemplate>
<%# Container.DataItem.Key %> =
<%# Container.DataItem.Value %><br />
</ItemTemplate>
</ASP:Repeater>
```

صيغة استخدام ربط البيانات للقيم المتكررة

عند ربط عنصر التحكم بقيمة وحيدة كحال الربط مع خاصة أو الربط مع القيمة المعادة من طريقة.

تحتوي مصادر البيانات المرتبطة بقيم متكررة على أكثر من حقل. ففي حال كان مصدر البيانات من النوع HashTable الذي يحتوي على مفتاح وقيمة أو كان من النوع DataReader، يجب أن نكون قادرين على تحديد العمود أو الحقل المراد ربط عنصر التحكم إليه.

تكون كثير من عناصر تحكم القوائم قادرة على إظهار أكثر من قيمة لكل عنصر في القائمة. فعلى سبيل المثال، نستطيع في العنصر <SELECT> استخدام قيمة لمحتوى العنصر <OPTION> وقيمة للخاصة VALUE لهذا العنصر.

إسقاط الحقول من مصدر البيانات على خصائص عنصر التحكم:

هناك طريقتان لإسقاط أو وصل حقل محدد من كل صف في مصدر البيانات إلى خصائص عنصر تحكم.

يعتمد اختيار الطريقة على عنصر التحكم الذي نقوم بربطه:

- إذا كان عنصر التحكم يدعم القوالب يمكننا التصريح عن قالب لكل صف سيتم إظهاره من قبل عنصر التحكم.
- إذا كان عنصر التحكم لا يدعم القوالب نستطيع إسناد الحقول في مصدر البيانات إلى خصائص عنصر التحكم بضبط الخصائص أثناء عمل البرنامج.

تكون عناصر التحكم في ASP.NET التي تدعم القوالب هي: Repeater، DataList و DataGrid وعليه، يمكننا لأي من هذه العناصر التصريح عن قالب ووضع تعليمات ربط البيانات ضمنه.

إسقاط أو وصل الحقول بصورة ديناميكية أثناء العمل

عند استخدام عناصر التحكم التي لا تدعم القوالب، لابد من ضبط قيم الخصائص أثناء العمل، تحدد هذه الخصائص اسم الحقل من مصدر البيانات الذي سيتم إظهاره في الخرج (الخاصة DataTextField) والحقل الذي سيعطي القيم غير الظاهرة لعنصر التحكم (الخاصة DataValueField).

```
<select id="MySelectList" runat="server" />
```

ثم نضيف في معالج الحدث Page_Load النص البرمجي:

```
MySelectList.DataSource = tabValues
MySelectList.DataValueField = "Key"
MySelectList.DataTextField = "Value"
MySelectList.DataBind()
```

ويعتبر عنصر التحكم DataGrid ذكياً بشكل كافي لتحديد الحقول من مصدر البيانات بصورة آلية وإظهار جميع القيم. يعمل هذا الحل عندما يكون مصدر البيانات هو Array، DataView، DataSet ، DataReader ولكن لا يعمل في حال كان مصدر البيانات هو HashTable.

إسقاط أو وصل الحقول بصورة ديناميكية أثناء العمل

عند استخدام عناصر التحكم التي لا تدعم القوالب، لابد من ضبط قيم الخصائص أثناء العمل، تحدد هذه الخصائص اسم الحقل من مصدر البيانات الذي سيتم إظهاره في الخرج (الخاصة DataTextField) والحقل الذي سيعطي القيم غير الظاهرة لعنصر التحكم (الخاصة DataValueField).

ويعتبر عنصر التحكم DataGrid ذكياً بشكل كافي لتحديد الحقول من مصدر البيانات بصورة آلية وإظهار جميع القيم. يعمل هذا الحل عندما يكون مصدر البيانات هو Array، DataView، DataSet ،

DataReader ولكن لا يعمل في حال كان مصدر البيانات هو HashTable.

تقييم التعبيرات بالطريقة Eval

يمكن أن تحتوي كتلة ربط البيانات في عناصر التحكم التي تستخدم القوالب أمثلة من التعبير التالي أو مشتقات عنها:

```
<%# Container.DataItem (“FieldName”) %>
```

يُعتبر استخدام طريقة Eval الخاصة بالغرض DataBinder أحد المشتقات الشائعة لتحديد قيمة ضمن مصدر البيانات، في حال احتوائه على أكثر من قيمة ضمن الصف. ويمنح خيار تنسيق القيمة لإظهارها.

هناك ثلاث وسائل نستطيع فيها استخدام الطريقة Eval:

- عند احتواء كل صف من القائمة ضمن مصدر البيانات على أكثر من قيمة (أكثر من حقل أو عمود). على سبيل المثال صف من DataView مرتبط بجدول من قاعدة بيانات أو غرض HashTable.
- عند الرغبة باستخدام غرض مختلف عن ذلك المرتبط بعنصر التحكم كمصدر للقيمة.
- عندما نريد تنسيق قيمة عند إظهارها. على سبيل المثال أخذ قيمة عددية وتنسيقها كعملة.

تعتبر الطريقة الأولى من الطرق السابقة امتداداً للصيغة المستخدمة سابقاً ولا تضيف عليها شيئاً سوى موضوع الأداء. في المثال التالي نريد القيمة من العمود المسمى BookTitle من الصف في غرض DataView:

```
<%# DataBinder.Eval (Container.DataItem,”BookTitle”) %>
```

تستخدم الطريقة Eval تقنية تدعى الربط المتأخر بعد تقييم التعبير مما يخلق حمل أعلى نسبة إلى الطريقة التقليدية المستخدمة.

أما الطريقة الثانية لاستخدام Eval فتُستَخدَم عندما نريد الربط إلى غرض غير معرف في الخاصة DataSource لعنصر التحكم. يسمح لنا ذلك بالتأشير إلى قيم معينة في هذا الغرض سيتم استخدامها في كل صف يتم إظهاره. فعلى سبيل المثال إذا كان لدينا غرض DataView باسم ObjCityData يحتوي معلومات حول المدن نستطيع تحديد قيمة العمود CityName في الصف الرابع من الغرض DataView باستخدام:

```
<%# DataBinder.Eval(ObjCityData, “[3].CityName”) %>
```

من أكثر تطبيقات الطريقة Eval فائدة، استخدامها لتنسيق القيم المراد إظهارها وهذا ما سنراه في الشريحة التالية.

يُعتبر استخدام طريقة Eval الخاصة بالغرض DataBinder أحد المشتقات الشائعة لتحديد قيمة ضمن مصدر البيانات، في حال احتوائه على أكثر من قيمة ضمن الصف. ويمنح خيار تنسيق القيمة لإظهارها.

هناك ثلاث وسائل نستطيع فيها استخدام الطريقة Eval:

- عند احتواء كل صف من القائمة ضمن مصدر البيانات على أكثر من قيمة (أكثر من حقل أو عمود). على سبيل المثال صف من DataView مرتبط بجدول من قاعدة بيانات أو غرض HashTable.
- عند الرغبة باستخدام غرض مختلف عن ذلك المرتبط بعنصر التحكم كمصدر للقيمة.
- عندما نريد تنسيق قيمة عند إظهارها. على سبيل المثال أخذ قيمة عددية وتنسيقها كعملة.

تعتبر الطريقة الأولى من الطرق السابقة امتداداً للصيغة المستخدمة سابقاً ولا تضيف عليها شيئاً سوى موضوع الأداء.
تستخدم الطريقة Eval تقنية تدعى الربط المتأخر بعد تقييم التعبير مما يخلق حمل أعلى نسبة إلى الطريقة التقليدية المستخدمة.

أما الطريقة الثانية لاستخدام Eval فتُستَخدَم عندما نريد الربط إلى غرض غير معرف في الخاصة DataSource لعنصر التحكم. يسمح لنا ذلك بالتأشير إلى قيم معينة في هذا الغرض سيتم استخدامها في كل صف يتم إظهاره.

من أكثر تطبيقات الطريقة Eval فائدة، استخدامها لتنسيق القيم المراد إظهارها وهذا ما سنراه في الشريحة التالية.

تنسيق القيم باستخدام طريقة EVal:

تأخذ الطريقة Eval ثلاث معاملات يكون الأخير منها وهو formatString الذي يحدد تنسيق الخرج، غير إجباري. فعلى سبيل المثال نستطيع تحديد محتوى الحقل PublicationField من مصدر البيانات الذي نود إظهاره بتنسيق تاريخ نظامي باستخدام الصيغة:

```
<%# DataBinder.Eval(Container.DataItem, "PublicationDate", "{0:D}")
%>
```

تكون النتيجة من الشكل "10 july 2006" (حسب الإعدادات الإقليمية للمخدم).

عند استخدامنا الطريقة Eval نستطيع الحصول على قيمة وحيدة في كل تعبير حيث يجب أن يكون الرقم الأول ضمن القوس صفراً دائماً (يستخدم لحجز مكان للمتغير الذي سيحمل القيمة المراد تنسيقها) ، أما الحرف التالي فهو الذي يعبر عن التنسيق المطلوب.

فيمايلي نسرد أهم التنسيقات المستخدمة:

التنسيقات العددية:

المحرف المستخدم للتنسيق	الوصف	مثال
C or c	Currency format	*$1,234.60*, *($28.15)*,
D or d	Decimal format	*205*, *17534*, *-65*
E or e	Scientific (exponential) format	*3.46E+21*, *-1.2e+3*, *3.003E-15*
F or f	Fixed-point format	*34.300*, *-0.230*
G or g	General format	Depends on actual value
N or n	Number format	*3,456.23*, *12.65*, *-1.534*
P or p	Percent format	*45.6%*, *-10%*
X or x	Hexadecimal format	*&H5f76*, *0x4528* (depends on actual value)

أما مع التاريخ فنستخدم:

المحرف المستخدم للتنسيق	الوصف	مثال
d	Short date	M/d/yyyy
D	Long date	dddd, MMMM dd, yyyy
f	Full (long date and short time)	dddd, MMMM dd, yyyy HH:mm aa
F	Full (long date and long time)	dddd, MMMM dd, yyyy HH:mm:ss aa
g	General (short date and short time)	M/d/yyyy HH:mm aa

M/d/yyyy HH:mm:ss aa	General (short date and long time)	G
MMMM dd	Month and day	M or m
ddd, dd MMM yyyy HH:mm:ssGMT	RFC1123 format	R or r
yyyy-MM-dd HH:mm:ss	ISO 8601 sortable using local time	s
HH:mm aa	Short time	t
yyyy-MM-dd HH:mm:s	ISO 8601 sortable using universal time	u
dddd, MMMM dd, yyyy HH:mm:ss aa	Universal sortable date/time	U
MMMM, yyyy	Year and month	Y or y

من الممكن أيضاً استخدام المحارف في الجدول التالي لإنشاء صورة لقيم رقمية، فعلى سبيل المثال يولد تنسيق "00#.##" عند التطبيق على الرقم 1.2345 الرقم "001.23". فإذا كنا نريد تحديد قيم موجبة أو سالبة وصفرية، فإننا نقوم بفصلها عن بعضها البعض باستخدام فاصلة منقوطة.

إذاً باسستخدام عبارة تنسيق من الشكل:

"00#.##;(00#.##);[0]"

نستطيع الحصول على

"(001.23)" للرقم 1.2345- و "[0]" للرقم صفر.

الوصف	محارف التنسيق
يقوم بإظهار صفر إذا لم يكن هناك قيمة أو يضيف صفراً أمام الرقم أو يحدد أصفار عشرية إضافية بعد الفاصلة العشرية.	0
يحل محله أي رقم ويهمل أي ظهور لهذا المحرف إذا لم يوجد رقم مرتبط	#
تظهر نقطة عشرية	.
تظهر محرف %	%
أو Scientificتقوم بتنسيق الخرج بالنمط Eباستخدام	E+0, E-0, e+0 or e-0
تقوم بإظهار المحارف بعدها كما هي بغض	\

	النظر عن كونها محارف تنسيق.
‘ أو “	يتم إظهار أي مجموعة محارف محتواة كما هي هي بغض النظر عن كونها محارف تنسيق.

تأخذ الطريقة Eval ثلاث معاملات يكون الأخير منها وهو formatString الذي يحدد تنسيق الخرج، غير إجباري.

1.7-8.16مثال عن ربط البيانات بقيمة مكررة. يجري استخدام ArrayList تم إنشاؤه وملؤه بالقيم ضمن معالج الحدث Page_Load:

Simple Repeated-Value Data Binding

HTML <**select**> element:
Microsoft

<**ASP:DropDownList**> control:
Microsoft

<**ASP:ListBox**> control:
Microsoft
Sun
IBM
Compaq

<**ASP:DataGrid**> control:

Item
Microsoft
Sun
IBM
Compaq
Oracle

<**ASP:Repeater**> control:
Microsoft Sun IBM Compaq Oracle

<**ASP:DataList**> control:
Microsoft
Sun
IBM
Compaq
Oracle

<**ASP:CheckBoxList**> control:
Microsoft
Sun
IBM
Compaq
Oracle

<**ASP:RadioButtonList**> control:
Microsoft
Sun
IBM
Compaq
Oracle

نلاحظ في جزء html الخاص بهذه الصفحة، أننا قمنا بالتصريح عن ثماني عناصر تحكم قوائم.

يمكننا رؤية ذلك فيما عدا Repeater و DataList.

كل ما نفعله هو تعريف عنصر التحكم نفسه:

```
HTML <b>&lt;select&gt;</b> element:<br />
<select id="MySelectList" runat="server" /><p />
<b>&lt;ASP:DropDownList&gt;</b> control:<br />
<ASP:DropDownList id="MyDropDown" runat="server" /><p />
<b>&lt;ASP:ListBox&gt;</b> control:<br />
<ASP:ListBox id="MyASPList" runat="server" /><p />
<b>&lt;ASP:DataGrid&gt;</b> control:<br />
<ASP:DataGrid id="MyDataGrid" runat="server" /><p />
<b>&lt;ASP:Repeater&gt;</b> control:<br />
<ASP:Repeater id="MyRepeater" runat="server">
<ItemTemplate>
<%# Container.DataItem %>
</ItemTemplate>
</ASP:Repeater><p />
<b>&lt;ASP:DataList&gt;</b> control:<br />
<ASP:DataList id="MyDataList" runat="server">
<ItemTemplate>
<%# Container.DataItem %>
</ItemTemplate>
</ASP:DataList><p />
<b>&lt;ASP:CheckBoxList&gt;</b> control:<br />
<ASP:CheckBoxList id="MyCheckList" runat="server" /><p />
<b>&lt;ASP:RadioButtonList&gt;</b> control:<br />
<ASP:RadioButtonList id="MyRadioList" runat="server" /><p />
```

أما بالنسبة إلى Repaeater و Datalist فهما يتطلبان تحديد الخرج الذي نريد للعناصر المكررة لذلك قمنا بإدراج <ItempTemplate> .

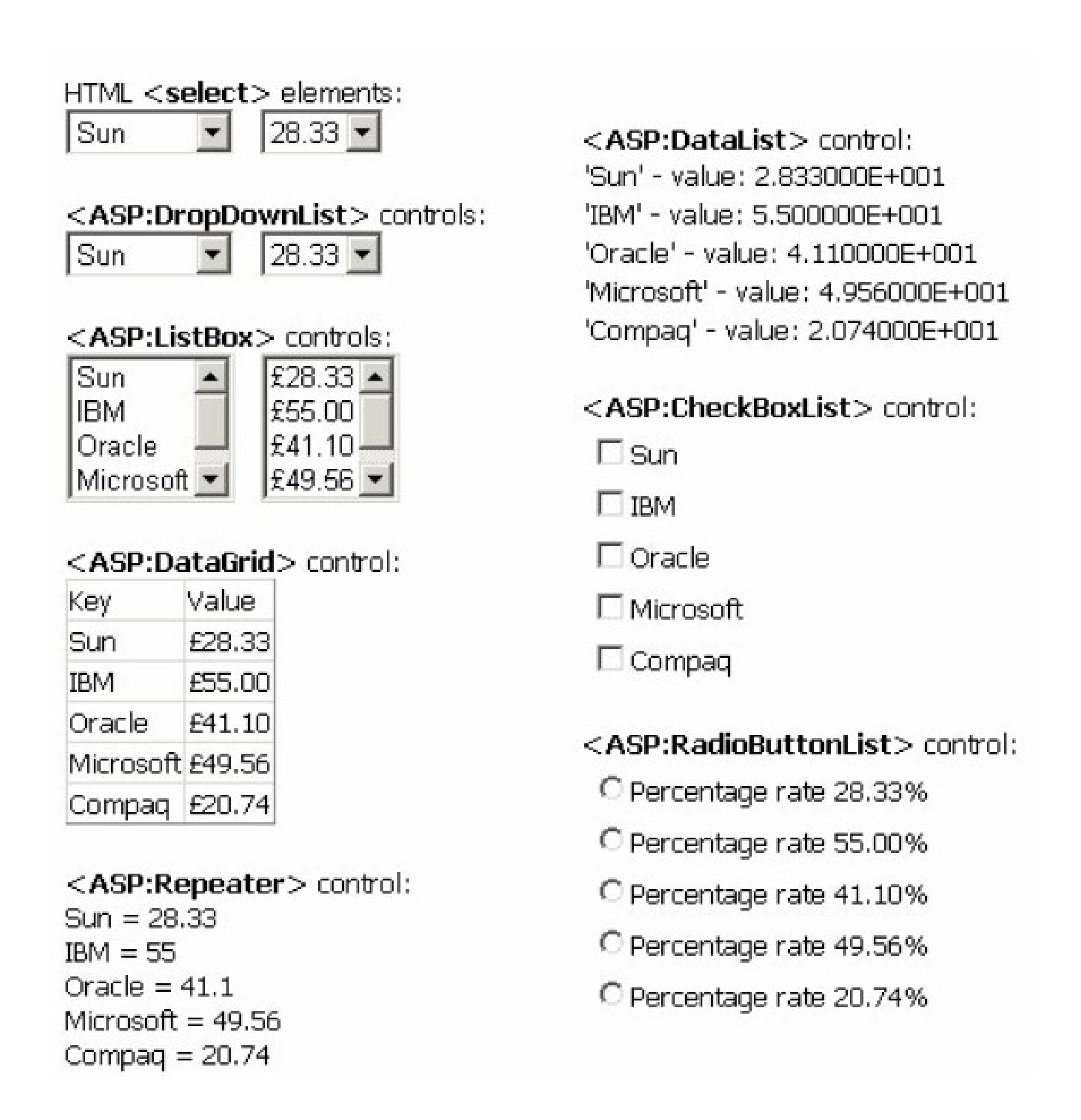
أياً كان ما ندرجه ضمن <ItempTemplate> فسيجري تنفيذه بعدد العناصر المكررة. نقوم في النص البرمجي بتحديد عنصر البيانات نفسه فقط ونضع القيمة لهذا العنصر ضمن ArrayList.

بالنسبة للنص البرمجي الخاص بمعالجة حدث تحميل الصفحة، نقوم ببساطة بإنشاء ArrayList ونملؤها بخمس قيم من سلاسل المحارف، ونقوم بإسناد الغرض ArrayList كقيمة للخاصة DataSource لعناصر التحكم. ثم نقوم بربط البيانات.

```
Sub Page_Load()
'create an ArrayList of values to bind to
Dim arrValues As New ArrayList(4)
arrValues.Add("Microsoft")
arrValues.Add("Sun")
arrValues.Add("IBM")
```

```
arrValues.Add("Compaq")
arrValues.Add("Oracle")
'set the DataSource propert of the controls to the array
MySelectList.DataSource = arrValues
MyDropDown.DataSource = arrValues
MyASPList.DataSource = arrValues
MyDataGrid.DataSource = arrValues
MyRepeater.DataSource = arrValues
MyDataList.DataSource = arrValues
MyCheckList.DataSource = arrValues
MyRadioList.DataSource = arrValues
'bind all the controls on the page
Page.DataBind()
End Sub
```

مثال عن ربط البيانات بقيمة مكررة يستخدم HashTable



في جزء html الخاص بهذه الصفحة نلاحظ أننا قمنا بالتصريح عن ثماني عناصر تحكم قوائم.

يمكننا رؤية ذلك فيما عدا Repeater و DataList.

كل ما نفعله هو تعريف عنصر التحكم نفسه:

```
HTML <b>&lt;select&gt;</b> elements:<br />
<select id="MySelectList1" runat="server" />  
<select id="MySelectList2" runat="server" /><p />
<b>&lt;ASP:DropDownList&gt;</b> controls:<br />
<ASP:DropDownList id="MyDropDown1" runat="server" />  
<ASP:DropDownList id="MyDropDown2" runat="server" /><p />
<b>&lt;ASP:ListBox&gt;</b> controls:<br />
<ASP:ListBox id="MyASPList1" runat="server" />  
<ASP:ListBox id="MyASPList2" runat="server" /><p />
<b>&lt;ASP:DataGrid&gt;</b> control:<br />
<ASP:DataGrid id="MyDataGrid" runat="server" AutoGenerateColumns="false">
```

```
<Columns>
<ASP:BoundColumn HeaderText="Key" DataField="Key" />
<ASP:BoundColumn HeaderText="Value" DataField="Value"
DataFormatString="{0:C}" />
</Columns>
</ASP:DataGrid><p />
<b>&lt;ASP:Repeater&gt;</b> control:<br />
<ASP:Repeater id="MyRepeater" runat="server">
<ItemTemplate>
<%# Container.DataItem.Key %> =
<%# Container.DataItem.Value %><br />
</ItemTemplate>
</ASP:Repeater><p />
<b>&lt;ASP:DataList&gt;</b> control:<br />
<ASP:DataList id="MyDataList" runat="server">
<ItemTemplate>
'<%# Container.DataItem.Key %>'- value:
<%# DataBinder.Eval(Container.DataItem, "Value", "{0:E}") %>
</ItemTemplate>
</ASP:DataList><p />
<b>&lt;ASP:CheckBoxList&gt;</b> control:<br />
<ASP:CheckBoxList id="MyCheckList" runat="server" /><p />
<b>&lt;ASP:RadioButtonList&gt;</b> control:<br />
<ASP:RadioButtonList id="MyRadioList" runat="server" /><p />
<ASP:DataGrid id="MyDataGrid" runat="server" AutoGenerateColumns="false">
<Columns>
<ASP:BoundColumn HeaderText="Key" DataField="Key" />
<ASP:BoundColumn HeaderText="Value" DataField="Value"
DataFormatString="{0:C}" />
</Columns>
</ASP:DataGrid>
```

و لربط عناصر Repeater و DataList نستخدم <itemTemplate>.

نلاحظ استخدامنا لعنصر البيانات Key وعنصر البيانات Value كوننا نتعامل مع HashTable.

```
<b>&lt;ASP:Repeater&gt;</b> control:<br />
<ASP:Repeater id="MyRepeater" runat="server">
<ItemTemplate>
<%# Container.DataItem.Key %> =
<%# Container.DataItem.Value %><br />
</ItemTemplate>
</ASP:Repeater><p />
<b>&lt;ASP:DataList&gt;</b> control:<br />
<ASP:DataList id="MyDataList" runat="server">
<ItemTemplate>
'<%# Container.DataItem.Key %>'- value:
<%# DataBinder.Eval(Container.DataItem, "Value", "{0:E}") %>
</ItemTemplate>
</ASP:DataList><p />
```

و أخيراً لملء غرض HashTable بالبيانات نستخدم معالج الحدث Page_Load:

```
Sub Page_Load()
'create a HashTable of values to bind to
Dim tabValues As New HashTable(5)
tabValues.Add("Microsoft", 49.56)
```

```
tabValues.Add("Sun", 28.33)
tabValues.Add("IBM", 55)
tabValues.Add("Compaq", 20.74)
tabValues.Add("Oracle", 41.1)
'first <select> displays the Keys in the HashTable
MySelectList1.DataSource = tabValues
MySelectList1.DataTextField = "Key"
'second one displays the Values in the HashTable
'and uses the Keys as the <option> values
MySelectList2.DataSource = tabValues
MySelectList2.DataValueField = "Key"
MySelectList2.DataTextField = "Value"
'same applies to ASP: controls, except here
'we can also specify the format of the Key
MyDropDown1.DataSource = tabValues
MyDropDown1.DataTextField = "Key"
MyDropDown2.DataSource = tabValues
MyDropDown2.DataValueField = "Key"
MyDropDown2.DataTextField = "Value"
MyDropDown2.DataTextFormatString = "{0:F}"
MyASPList1.DataSource = tabValues
MyASPList1.DataTextField = "Key"
MyASPList2.DataSource = tabValues
MyASPList2.DataValueField = "Key"
MyASPList2.DataTextField = "Value"
MyASPList2.DataTextFormatString = "{0:C}"
```

و نحدد لعناصر تحكم Repeater و DataList مصدر البيانات فقط لأننا حددنا ضمن ItemTemplate كيف سيتم إظهار المحتويات.

```
MyDataGrid.DataSource = tabValues
MyRepeater.DataSource = tabValues
MyDataList.DataSource = tabValues
```

أما بالنسبة لعناصر التحكم CheckBoxList و RadioButtonList فيكفي تحديد الخصائص DataValueField و DataTextField لها:

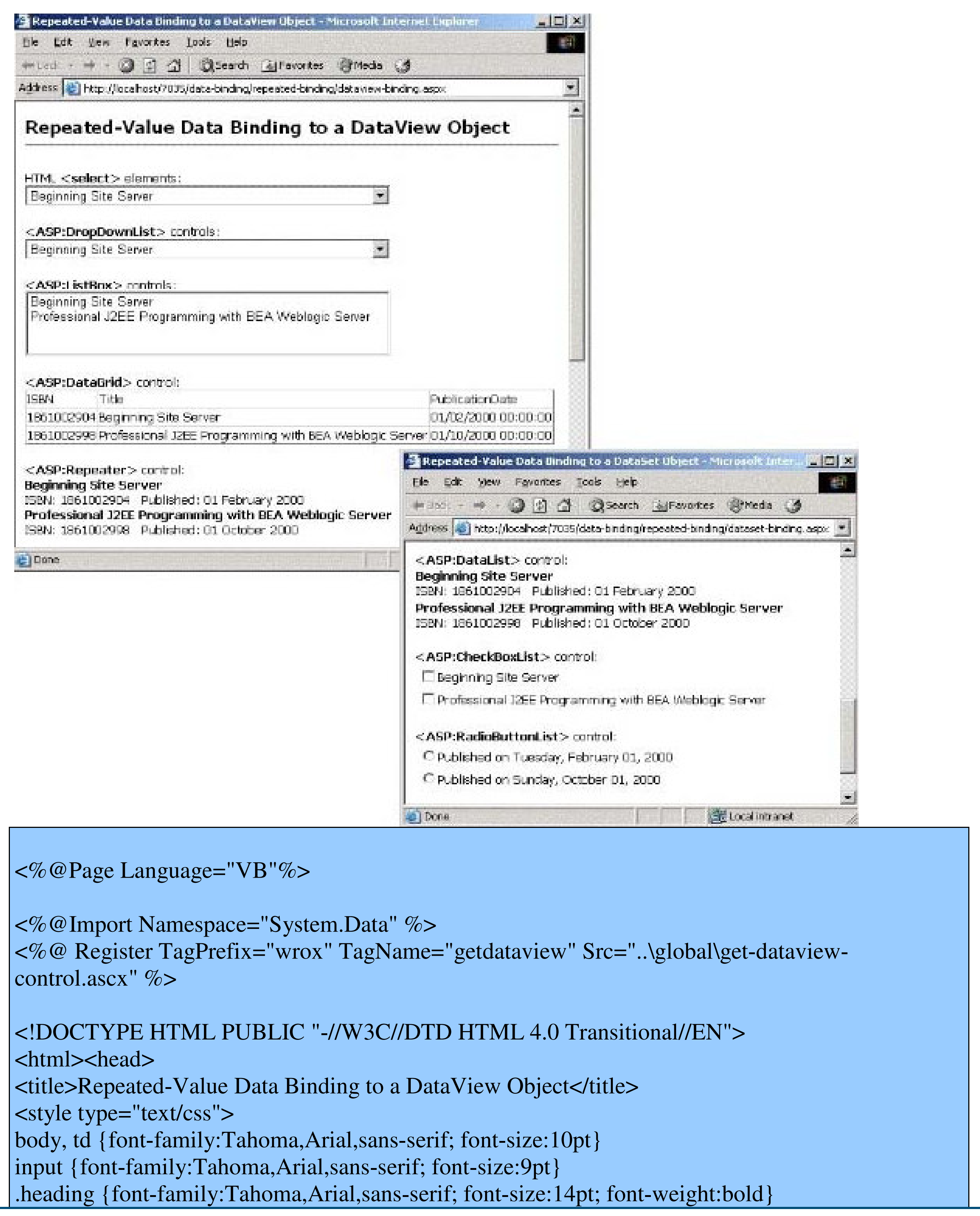
```
'in the CheckboxList we'll display the Title and
'use the Value as the control value
MyCheckList.DataSource = tabValues
MyCheckList.DataValueField = "Value"
MyCheckList.DataTextField = "Key"
'in the RadioList we'll display and format the
'Value and use the Key as the control value
MyRadioList.DataSource = tabValues
MyRadioList.DataValueField = "Key"
MyRadioList.DataTextField = "Value"
MyRadioList.DataTextFormatString = "Percentage rate {0:F}%"
```

و بالطبع يجب ألا ننسى عملية ربط جميع العناصر في الصفحة

```
Page.DataBind() 'bind all the controls on the page
End Sub
```

مثال ربط قيمة متكررة إلى غرض DataView:

في هذا المثال سنستعرض ربط البيانات إلى غرض Dataview. سنتعامل مع هذا الغرض فقط كحاوية لمجموعة من الصفوف لأننا سنؤجل الحديث عن الاتصال مع قواعد البيانات وكيفية الحصول على البيانات منها إلى جلسة مخصصة لهذا الغرض.




```
<%@Page Language="VB"%>

<%@Import Namespace="System.Data" %>
<%@ Register TagPrefix="wrox" TagName="getdataview" Src="..\global\get-dataview-
control.ascx" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Repeated-Value Data Binding to a DataView Object</title>
<style type="text/css">
body, td {font-family:Tahoma,Arial,sans-serif; font-size:10pt}
input {font-family:Tahoma,Arial,sans-serif; font-size:9pt}
.heading {font-family:Tahoma,Arial,sans-serif; font-size:14pt; font-weight:bold}
```

```
.subhead {font-family:Tahoma,Arial,sans-serif; font-size:12pt; font-weight:bold; padding-
bottom:5px}
.cite {font-family:Tahoma,Arial,sans-serif; font-size:8pt}
</style></head>
<body bgcolor="#ffffff">
<span class="heading">Repeated-Value Data Binding to a DataView Object</span><hr />
<!----------------------------------------------------------------------->

<%'-- insert the control that creates the DataSet --%>
<wrox:getdataview id="ctlDataView" runat="server"/>

<form runat="server">

HTML <b>&lt;select&gt;</b> elements:<br />
<select id="MySelectList" runat="server" /><p />

<b>&lt;ASP:DropDownList&gt;</b> controls:<br />
<ASP:DropDownList id="MyDropDown" runat="server" /><p />

<b>&lt;ASP:ListBox&gt;</b> controls:<br />
<ASP:ListBox id="MyASPList" runat="server" /><p />

<b>&lt;ASP:DataGrid&gt;</b> control:<br />
<ASP:DataGrid id="MyDataGrid" runat="server" /><p />

<b>&lt;ASP:Repeater&gt;</b> control:<br />
<ASP:Repeater id="MyRepeater" runat="server">
<ItemTemplate>
<div>
<b><%# Container.DataItem("Title") %></b><br />
ISBN: <%# Container.DataItem("ISBN") %>  
Published: <%# DataBinder.Eval(Container.DataItem, "PublicationDate", "{0:D}") %>
</div>
</ItemTemplate>
</ASP:Repeater><p />
<b>&lt;ASP:DataList&gt;</b> control:<br />
<ASP:DataList id="MyDataList" runat="server">
<ItemTemplate>
<b><%# Container.DataItem("Title") %></b><br />
ISBN: <%# Container.DataItem("ISBN") %>  
Published: <%# DataBinder.Eval(Container.DataItem, "PublicationDate", "{0:D}") %>
</ItemTemplate>
</ASP:DataList><p />

<b>&lt;ASP:CheckBoxList&gt;</b> control:<br />
<ASP:CheckBoxList id="MyCheckList" runat="server" /><p />

<b>&lt;ASP:RadioButtonList&gt;</b> control:<br />
<ASP:RadioButtonList id="MyRadioList" runat="server" /><p />

</form>
```

```
<!------------------------------------------------------------------------->

<script language="vb" runat="server">

Sub Page_Load()

'get connection string from web.config
Dim strConnect As String = ConfigurationSettings.AppSettings("DsnWroxBooksOleDb")

'create a SQL statement to select some rows from the database
Dim strSelect As String
strSelect = "SELECT * FROM BookList WHERE ISBN LIKE '07645438%'"

'create a variable to hold an instance of a DataView object
Dim objDataView As DataView

'get DataView from get-dataview-control.ascx user control
objDataView = ctlDataView.GetDataView(strConnect, strSelect)

If IsNothing(objDataView) Then Exit Sub

'set the DataSource property of the controls

'<select> list displays values from the Title column
'and uses the ISBN as the <option> values
MySelectList.DataSource = objDataView
MySelectList.DataValueField = "ISBN"
MySelectList.DataTextField = "Title"

'do same with ASP: list controls
MyDropDown.DataSource = objDataView
MyDropDown.DataValueField = "ISBN"
MyDropDown.DataTextField = "Title"

MyASPList.DataSource = objDataView
MyASPList.DataValueField = "ISBN"
MyASPList.DataTextField = "Title"

'a DataGrid can figure out the columns in the DataView
'by itself, so we just set the DataSource property
MyDataGrid.DataSource = objDataView

'the Repeater and DataList require <ItemTemplate> entries
'that specify the columns - this is done within the element
'definition earlier in the page
MyRepeater.DataSource = objDataView
MyDataList.DataSource = objDataView

'in the CheckboxList we'll display the Title and
'use the Value as the control value
```

```
MyCheckList.DataSource = objDataView
MyCheckList.DataValueField = "ISBN"
MyCheckList.DataTextField = "Title"

'in the RadioList we'll display and format the
'Value and use the Key as the control value
MyRadioList.DataSource = objDataView
MyRadioList.DataValueField = "ISBN"
MyRadioList.DataTextField = "PublicationDate"
MyRadioList.DataTextFormatString = "Published on {0:dddd, MMMM dd, yyyy}"

'finally, bind all the controls on the page
Page.DataBind()

End Sub

</script>

<!-------------------------------------------------------------------------->
</body>
</html>
```

مثال ربط قيمة متكررة إلى غرض DataReader:

في هذا المثال سنستعرض ربط البيانات إلى غرض DataReader. سنتعامل مع هذا الغرض فقط كحاوية لمجموعة من الصفوف لأننا سنؤجل الحديث عن الاتصال مع قواعد البيانات وكيفية الحصول على البيانات منها إلى جلسة مخصصة لهذا الغرض.

Repeated-Value Data Binding to a DataReader Object

<**ASP:DataGrid**> control:

ISBN	Title	PublicationDate
1861003013	Professional Linux Programming	01/09/2000 00:00:00
186100303X	Professional Symbian Programming	01/02/2000 00:00:00
1861003064	Beginning Visual Basic SQL Server 7.0	01/01/2000 00:00:00
1861003129	XSLT Programmers Reference	01/04/2000 00:00:00
1861003145	Beginning Perl	01/06/2000 00:00:00
1861003218	Alex Homer's Professional ASP Web Techniques	01/02/2000 00:00:00
1861003234	ASP 3.0 Programmer's Reference	01/03/2000 00:00:00
1861003242	Professional ADO 2.5 RDS Programming With ASP 3.0	01/01/2000 00:00:00
1861003293	Professional BizTalk	01/01/2001 00:00:00
1861003323	Professional Visual Basic 6 XML	01/03/2000 00:00:00
1861003412	Beginning XML	01/06/2000 00:00:00
1861003439	Beginning XHTML	01/03/2000 00:00:00
1861003552	Professional Jini	01/08/2000 00:00:00
1861003579	Professional XSL	01/04/2001 00:00:00
1861003587	Professional XML Databases	01/12/2000 00:00:00
1861003625	Professional Java Server Pages	01/05/2000 00:00:00
1861003641	Professional Java Custom UI Components	01/08/2001 00:00:00
1861003668	Beginning Java 2 - JDK 1.3 Version	01/03/2000 00:00:00
1861003730	Beginning PHP4	01/10/2000 00:00:00
1861003757	Professional Java Web Services	01/01/2002 00:00:00
1861003811	Beginning GTK+/GNOME Programming	01/03/2000 00:00:00
186100382X	Professional Java Programming	01/12/2000 00:00:00
1861003897	Professional Java Mobile Programming	01/07/2001 00:00:00
1861003927	Professional ASP Data	01/10/2000 00:00:00
1861003986	Beginning E-Commerce With Visual Basic, ASP, SQL Server 7.0 And MTS	01/03/2000 00:00:00

```
<%@Page Language="VB"%>

<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.OleDb" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Repeated-Value Data Binding to a DataReader Object</title>
<style type="text/css">
body, td {font-family:Tahoma,Arial,sans-serif; font-size:10pt}
input {font-family:Tahoma,Arial,sans-serif; font-size:9pt}
.heading {font-family:Tahoma,Arial,sans-serif; font-size:14pt; font-weight:bold}
.subhead {font-family:Tahoma,Arial,sans-serif; font-size:12pt; font-weight:bold; padding-
bottom:5px}
.cite {font-family:Tahoma,Arial,sans-serif; font-size:8pt}
```

```
</style></head>
<body bgcolor="#ffffff">
<span class="heading">Repeated-Value Data Binding to a DataReader Object</span><hr />
<!------------------------------------------------------------------------->

<div id="outError" runat="server" />

<!-- Unlike the DataView we can only bind a object DataReader to one control -->
<!-- after it has bound the data the reader is at the end of the source rowset -->
<b>&lt;ASP:DataGrid&gt;</b> control:<br />
<ASP:DataGrid id="MyDataGrid" runat="server" /><p />

<!------------------------------------------------------------------------->

<script language="vb" runat="server">

Sub Page_Load()

'get connection string from web.config
Dim strConnect As String = ConfigurationSettings.AppSettings("DsnWroxBooksOleDb")

'create a SQL statement to select some rows from the database
Dim strSelect As String
strSelect = "SELECT * FROM BookList WHERE ISBN LIKE '07645437%'"

'create a variable to hold an instance of a DataReader object
Dim objDataReader As OleDbDataReader

Try

'create a new Connection object using the connection string
Dim objConnect As New OleDbConnection(strConnect)

'open the connection to the database
objConnect.Open()

'create a new Command using the connection object and select statement
Dim objCommand As New OleDbCommand(strSelect, objConnect)

'execute the SQL statement against the command to get the DataReader
objDataReader = objCommand.ExecuteReader()

Catch objError As Exception

'display error details
outError.InnerHtml = "<b>* Error while accessing data</b>.<br />" _
& objError.Message & "<br />" & objError.Source & "<p />"
Exit Sub  ' and stop execution

End Try
```

```
'set the DataSource property of the control
'a DataGrid can figure out the columns in the DataReader
'by itself, so we just set the DataSource property
MyDataGrid.DataSource = objDataReader
MyDataGrid.DataBind()  'and bind the control

End Sub

</script>

<!------------------------------------------------------------------------->
</body>
</html>
```

العمل مع الأنماط والقوالب

رأينا خلال الجلسة الماضية وخلال هذه الجلسة كيف أن عملية ربط البيانات هي عملية غاية في السهولة ورأينا كيف توفر الجهد وتقلل الحاجة إلى كتابة نصوص برمجية طويلة.

لكن ما سنتطرق إليه في هذا الجزء هو كيفية إخراج البيانات بالمظهر المطلوب . إذ يمكن الوصول إلى هذا الغرض بثلاث طرق:

- إضافة أنماط CSS إلى عنصر التحكم إما مباشرةً، أو عن طريق التأشيرة <Style> في الصفحة، أو بإسناد قيمة إلى الخاصة Style لعنصر التحكم.
- إنشاء قالب يحدد طريقة الإظهار في مقاطع مختلفة من خرج عنصر التحكم.
- استخدام مزيج من الطريقتين السابقتين.

إضافة أنماط CSS إلى عنصر التحكم

يمكن التحكم بالنمط مباشرة عن طريق استخدام التأشيرة style:

```
<style type="text/css">
body, td {font-family:Tahoma,Arial,sans-serif; font-size:10pt}
input {font-family:Tahoma,Arial,sans-serif; font-size:9pt}
</style>
```

في المثال السابق ستأخذ جميع العناصر التي تستخدم <input> إضافة إلى كل عناصر تحكم نماذج الوب بالتنسيق المحدد في input.

استخدام الخاصة Style لعناصر التحكم:

تتوي معظم عناصر تحكم القوائم المصممة للربط مع البيانات على مجموعة من الخصائص التي يمكنها تعديل الأنماط المستخدمة في الصفحة. يمكن استخدام هذه الأخيرة لتغيير مظهر العناصر في الصفحة باستثناء عنصر التحكم Repeater.

بعض الخصائص التي يمكن ضبطها تظهر في الجدول أدناه:

الخاصة	الوصف
BackColor, BackImageUrl	تقوم بضبط مظهر الخلفية لعنصر التحكم
BorderStyle, BorderColor, BorderWidth	تقوم بضبط إطار عنصر التحكم
GridLines, CellPadding, CellSpacing	تقوم بضبط مظهر الخلية
Font-Name, Font-Size, Font-Bold	تحدد نمط النص في عنصر التحكم
HeaderStyle, ItemStyle, FooterStyle AlternatingItemStyle	تحدد نمط الأجزاء المختلفة من خرج عنصر التحكم كالرأس والتذييل والعناصر .

رأينا خلال الجلسة الماضية وخلال هذه الجلسة كيف أن عملية ربط البيانات هي عملية غاية في السهولة ورأينا كيف توفر الجهد وتقلل الحاجة إلى كتابة نصوص برمجية طويلة.

لكن ما سنتطرق إليه في هذا الجزء هو كيفية إخراج البيانات بالمظهر المطلوب . إذ يمكن الوصول إلى هذا الغرض بثلاث طرق:

- إضافة أنماط CSS إلى عنصر التحكم إما مباشرةً، أو عن طريق التأشيرة <Style> في الصفحة، أو بإسناد قيمة إلى الخاصة Style لعنصر التحكم.
- إنشاء قالب يحدد طريقة الإظهار في مقاطع مختلفة من خرج عنصر التحكم.
- استخدام مزيج من الطريقتين السابقتين.

إضافة أنماط CSS إلى عنصر التحكم

يمكن التحكم بالنمط مباشرة عن طريق استخدام التأشيرة style.

استخدام الخاصة Style لعناصر التحكم:

تتوي معظم عناصر تحكم القوائم المصممة للربط مع البيانات على مجموعة من الخصائص التي يمكنها تعديل الأنماط المستخدمة في الصفحة. يمكن استخدام هذه الأخيرة لتغيير مظهر العناصر في الصفحة باستثناء عنصر التحكم Repeater.

العمل مع الأنماط والقوالب

مثال على استخدام Css للتحكم بطريقة إظهار الخرج لعناصر التحكم:

الشكل التالي هو الخرج للنص البرمجي أدناه:

Using CSS to Add Style to a DataGrid

ISBN	Title	PublicationDate
1861003218	Alex Homer's Professional ASP Web Techniques	01/02/2000 00:00:00
1861003234	ASP 3.0 Programmer's Reference	01/03/2000 00:00:00
1861003242	Professional ADO 2.5 RDS Programming With ASP 3.0	01/01/2000 00:00:00
1861003927	Professional ASP Data	01/10/2000 00:00:00
1861003986	Beginning E-Commerce With Visual Basic, ASP, SQL Server 7.0 And MTS	01/03/2000 00:00:00
1861004028	Professional ASP XML	01/06/2000 00:00:00
1861004397	Professional C# Web Services: Building .NET Web Services with ASP.NET and .NET Remoting	01/12/2001 00:00:00
1861004753	A Preview of ASP+	01/07/2000 00:00:00
1861005040	Beginning ASP.NET	01/08/2001 00:00:00
1861005229	Beginning ASP.NET Mobile Controls	01/12/2001 00:00:00
186100530X	ASP.NET Programmer's Reference	01/09/2001 00:00:00
1861005458	Professional ASP.NET Web Services	01/11/2001 00:00:00
1861005644	Professional ASP.NET Server Controls	01/02/2002 00:00:00
1861006152	Beginning ASP.NET using C#	01/11/2001 00:00:00
1861007035	Professional ASP.NET 2nd Edition	01/02/2002 00:00:00

النص البرمجي:

```
<%@Page Language="VB"%>

<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.OleDb" %>

<%@ Register TagPrefix="wrox" TagName="getdataview" Src="..\global\get-dataview-
control.ascx" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Using CSS to Add Style to a DataGrid</title>
<style type="text/css">
body {font-family:Tahoma,Arial,sans-serif; font-size:10pt}
```

```
input {font-family:Tahoma,Arial,sans-serif; font-size:9pt}
.heading {font-family:Tahoma,Arial,sans-serif; font-size:14pt; font-weight:bold}
.subhead {font-family:Tahoma,Arial,sans-serif; font-size:12pt; font-weight:bold; padding-
bottom:5px}
.cite {font-family:Tahoma,Arial,sans-serif; font-size:8pt}
</style></head>
<body bgcolor="#ffffff">
<span class="heading">Using CSS to Add Style to a DataGrid</span><hr />
<!---------------------------------------------------------------------------->

<div id="outError" runat="server" />

<ASP:DataGrid id="MyDataGrid" runat="server"
ShowHeader = "True"
ShowFooter = "False"
BackColor = "darkgray"
BackImageUrl = "background.gif"
ToolTip = "A List of Wrox Books"
GridLines = "None"
BorderStyle = "Solid"
BorderColor = "black"
BorderWidth = "3"
CellPadding = "2"
CellSpacing = "2"
Font-Name = "Comic Sans MS"
Font-Size = "10pt"
Font-Bold = "True" >
<HeaderStyle ForeColor = "blue" />
<ItemStyle ForeColor = "red" />
<AlternatingItemStyle ForeColor = "green" />
</ASP:DataGrid>

<!---------------------------------------------------------------------------->

<script language="vb" runat="server">

Sub Page_Load()

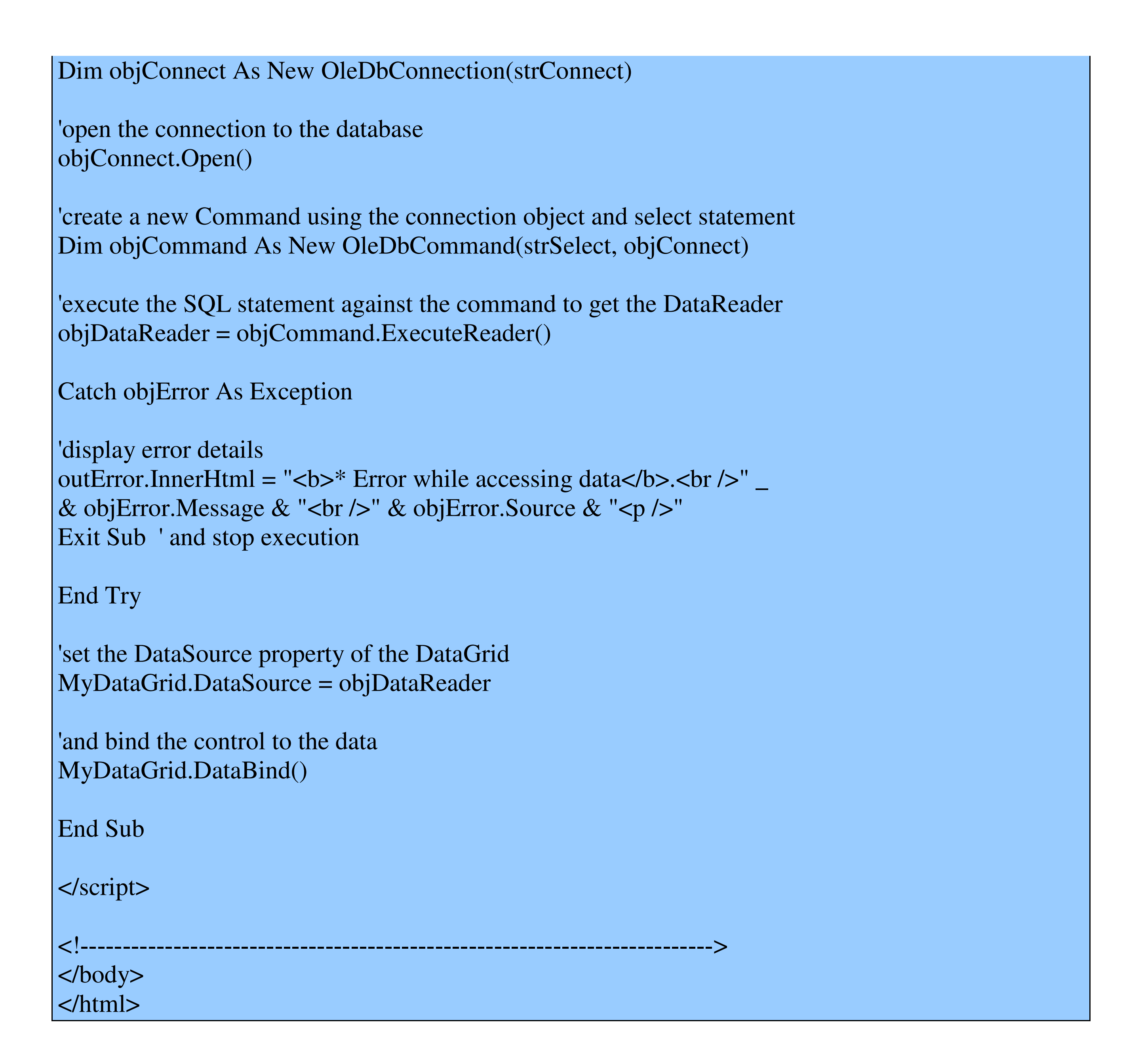
'get connection string from web.config
Dim strConnect As String = ConfigurationSettings.AppSettings("DsnWroxBooksOleDb")

'create a SQL statement to select some rows from the database
Dim strSelect As String
strSelect = "SELECT * FROM BookList WHERE Title LIKE '%ASP%'"

'create a variable to hold an instance of a DataReader object
Dim objDataReader As OleDbDataReader

Try

'create a new Connection object using the connection string
```



```
Dim objConnect As New OleDbConnection(strConnect)

'open the connection to the database
objConnect.Open()

'create a new Command using the connection object and select statement
Dim objCommand As New OleDbCommand(strSelect, objConnect)

'execute the SQL statement against the command to get the DataReader
objDataReader = objCommand.ExecuteReader()

Catch objError As Exception

'display error details
outError.InnerHtml = "<b>* Error while accessing data</b>.<br />" _
& objError.Message & "<br />" & objError.Source & "<p />"
Exit Sub  ' and stop execution

End Try

'set the DataSource property of the DataGrid
MyDataGrid.DataSource = objDataReader

'and bind the control to the data
MyDataGrid.DataBind()

End Sub

</script>

<!---------------------------------------------------------------------->
</body>
</html>
```

استخدام القوالب مع عناصر التحكم المرتبطة

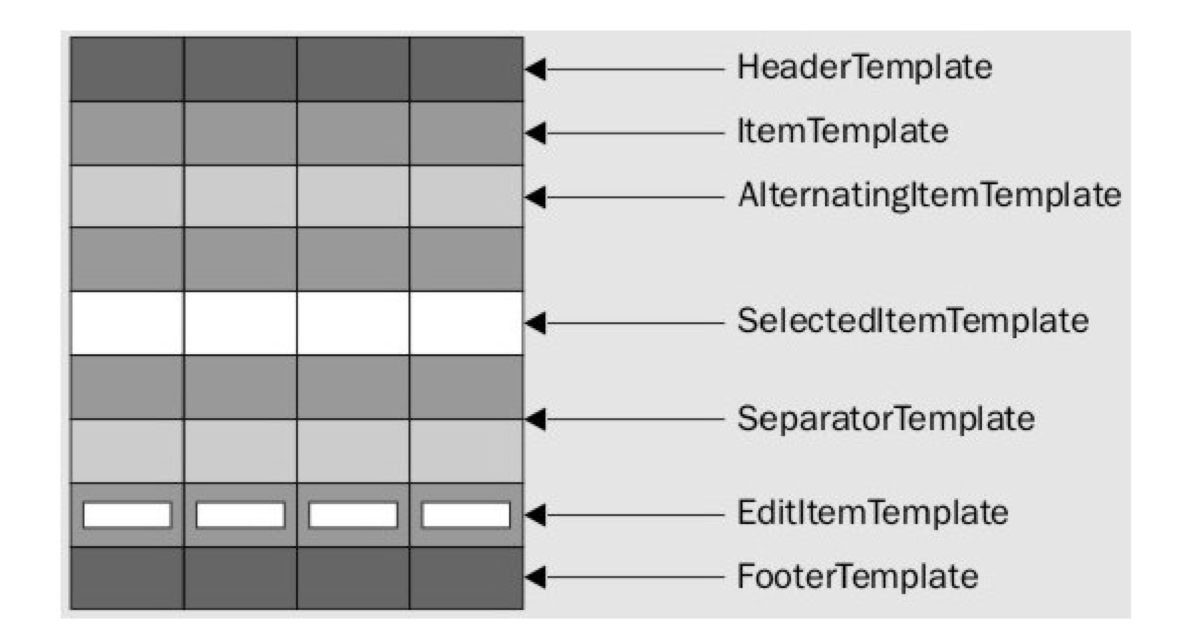
تتلخص الطريقة الثانية للتحكم بالمظهر الخاص لخرج عناصر تحكم القوائم في إضافة قوالب إلى تعريف عنصر التحكم.

يمكن للقوالب أن تقوم بتحديد الأعمدة التي سيتم إظهارها وكيف سيتم إظهار القيم.

تقبل ثلاثة من عناصر التحكم استخدام القوالب وهي العناصر: DataGrid،DataList،Repeater

يمكن أن نستخدم مع هذه العناصر مجموعة من القوالب التي تحدد طريقة الإظهار والمحتوى الخاص بكل جزء من الخرج.

فيمايلي رسم يبين المجموعة المتاحة من القوالب:



تفسر أسماء القوالب نفسها ولكن يبقى القالب تفسير SelectedItemTemplate والقالبEditItemTemplate حيث يحدد الأول طريقة إظهار العناصر التي يجري اختيارها.
أما الثاني فهو مخصص لتحديد طريقة إظهار العناصر التي يجري تحرير قيمها.

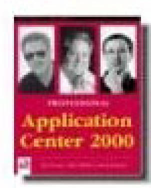
تتلخص الطريقة الثانية للتحكم بالمظهر الخاص لخرج عناصر تحكم القوائم في إضافة قوالب إلى تعريف عنصر التحكم.
يمكن للقوالب أن تقوم بتحديد الأعمدة التي سيتم إظهارها وكيف سيتم إظهار القيم.

تقبل ثلاثة من عناصر التحكم استخدام القوالب وهي العناصر :DataGrid،DataList،Repeater
يمكن أن نستخدم مع هذه العناصر مجموعة من القوالب التي تحدد طريقة الإظهار والمحتوى الخاص بكل جزء من الخرج.

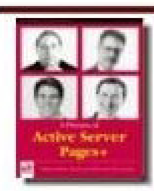
مثال عن استخدام قالب بسيط مع عنصر تحكم Repeater:

Using a Simple Template with a Repeater Control

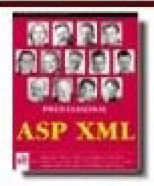
Some of the Latest Wrox Press Books



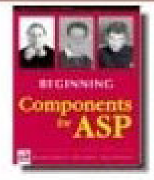
Professional Application Center 2000
ISBN: 1861004478 Published: 01/03/2001
This book takes you through the complete process of setting up and creating Web clusters and component clusters, adapting your applications to work in a clustered scenario, and how you can administer and monitor them.



A Preview of ASP+
ISBN: 1861004753 Published: 01/07/2000
Active Server Pages (ASP.NET) makes it easier, faster and less error-prone to write your own Web applications and dynamic Web pages. In a lot of cases, there is actually less code to write - and sometime none at all.



Professional ASP XML Programming
ISBN: 1861004028 Published: 01/05/2000
As ASP is probably the best server-side coding platform for Windows-based systems, it's an obvious choice for use with XML. This book looks at all of the XML-related issues you will face when working in this environment.



Beginning Components for ASP
ISBN: 1861002882 Published: 01/03/2000
If you use ASP, you're only getting half the power of this exciting programming environment if you don't build and use components. This book shows you how to get started, with plenty of tips, tricks and useful information.

النص البرمجي:

```
<%@Page Language="VB"%>

<%@Import Namespace="System.Data" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Using a Simple Template with a Repeater Control</title>
<style type="text/css">
body, td {font-family:Tahoma,Arial,sans-serif; font-size:10pt}
input {font-family:Tahoma,Arial,sans-serif; font-size:9pt}
.heading {font-family:Tahoma,Arial,sans-serif; font-size:14pt; font-weight:bold}
.subhead {font-family:Tahoma,Arial,sans-serif; font-size:12pt; font-weight:bold; padding-
bottom:5px}
.cite {font-family:Tahoma,Arial,sans-serif; font-size:8pt}
```

```
.rHead {font-family:Lucida Handwriting,Comic Sans MS,Tahoma,Arial;
font-size:14pt; font-weight:bold; padding:8px; color:green}
.rItem {font-family:Lucida Handwriting,Comic Sans MS,Tahoma,Arial,sans-serif;
font-size:10pt}
.rFoot {font-family:Tahoma,Arial; font-size:8pt; padding:8px; color:darkgray}

</style></head>
<body bgcolor="#ffffff">
<span class="heading">Using a Simple Template with a Repeater Control</span><hr />
<!------------------------------------------------------------------------->

<ASP:Repeater id="MyRepeater" runat="server">

<HeaderTemplate>
<div class="rHead">
Some of the Latest Wrox Books from Wiley<br />
<img src="images/redrule.gif">
</div>
</HeaderTemplate>

<ItemTemplate>
<div class="rItem">
<img src="images/<%# Container.DataItem("ImageURL") %>"
align="left" hspace="10" />
<b><%# Container.DataItem("Title") %></b><br />
ISBN: <%# Container.DataItem("ISBN") %>  
Published: <%# DataBinder.Eval(Container.DataItem, _
"PublicationDate", "{0:d}") %><br />
<%# Container.DataItem("Precis") %>
</div><br clear="all" />
</ItemTemplate>

<SeparatorTemplate>
<img src="images/redrule.gif">
</SeparatorTemplate>

<FooterTemplate>
<img src="images/redrule.gif">
<div class="rFoot">
For more information visit <a href="http://www.wrox.com">http://www.wrox.com</a>
</div>
</FooterTemplate>

</ASP:Repeater><p />

<!------------------------------------------------------------------------->

<script language="vb" runat="server">

Sub Page_Load()
```

```
'create a new empty DataTable object
Dim objTable As New DataTable("NewTable")

'define four columns (fields) within the table
objTable.Columns.Add("ISBN", System.Type.GetType("System.String"))
objTable.Columns.Add("Title", System.Type.GetType("System.String"))
objTable.Columns.Add("PublicationDate", System.Type.GetType("System.DateTime"))
objTable.Columns.Add("ImageURL", System.Type.GetType("System.String"))
objTable.Columns.Add("Precis", System.Type.GetType("System.String"))

'declare a variable to hold a DataRow object
Dim objDataRow As DataRow

'create a new DataRow object instance in this table
objDataRow = objTable.NewRow()

'and fill in the values
objDataRow("ISBN") = "0764544020"
objDataRow("Title") = "Beginning Access 2002 VBA"
objDataRow("PublicationDate") = "February 2003"
objDataRow("ImageURL") = "4020.gif"
objDataRow("Precis") = "Access 2002 is the core database application within " _
& "the Office XP suite. This book focuses on the programming language that " _
& "underlies Access 2002 and all the other office products."
objTable.Rows.Add(objDataRow)

objDataRow = objTable.NewRow()
objDataRow("ISBN") = "0764543636"
objDataRow("Title") = "Beginning Active Server Pages 3.0"
objDataRow("PublicationDate") = "July 2000"
objDataRow("ImageURL") = "3636.gif"
objDataRow("Precis") = "This book is for beginners who have no previous experience " _
& "of ASP, C#, XML, object-oriented programming, or the .NET Framework. A little " _
& "knowledge of HTML is useful, but by no means essential as all the concepts that " _
& "you need in order to create dynamic ASP.NET web sites are presented and explained in full."
objTable.Rows.Add(objDataRow)

objDataRow = objTable.NewRow()
objDataRow("ISBN") = "0764543695"
objDataRow("Title") = "Beginning ASP.NET 1.0 with Visual Basic.NET"
objDataRow("PublicationDate") = "February 2003"
objDataRow("ImageURL") = "3695.gif"
objDataRow("Precis") = "This book is for beginners who have no previous experience of " _
& "ASP, VB, XML, object-oriented programming, or the .NET Framework. All the concepts " _
& "you need in order to create dynamic ASP.NET web sites are presented and explained in full."
objTable.Rows.Add(objDataRow)

objDataRow = objTable.NewRow()
objDataRow("ISBN") = "0764543962"
objDataRow("Title") = "Professional ASP.NET 1.0, Special Edition"
```

```
objDataRow("PublicationDate") = "February 2002"
objDataRow("ImageURL") = "3962.gif"
objDataRow("Precis") = "This book is for people that have a solid understanding of ASP " _
& "and are familiar with VB or C-based syntax. It will show you how to develop sophisticated " _
& "ASP.NET applications using the .NET Framework with its comprehensive and in-depth guide " _
& "to this exciting new technology."
objTable.Rows.Add(objDataRow)

'assign the DataTable's DefaultView object to the Repeater control
MyRepeater.DataSource = objTable.DefaultView
MyRepeater.DataBind()  'and bind (display) the data

End Sub

</script>

<!---------------------------------------------------------------------->
</body>
</html>
```

الفصل التاسع والعاشر

عنوان الموضوع:

إدراة الحالة وأحداث تطبيقات الوب

الكلمات المفتاحية:

الحالة ، الجلسة، التطبيق، حدث.

ملخص:

يعتبر حفظ وإدارة معلومات الحالة عملية أساسية في بيئة الوب. سنتعرف في هذه الجلسة على الآلية التي تتعامل بها ASP.NET مع الحالة كما سنتعرف على الأحداث الخاصة بتطبيقات الوب.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- مفهوم إدارة الحالة
- أغراض الحالة في ASP.NET
- أحداث التطبيق في ASP.NET

إدارة الحالة في تطبيقات ASP.NET

تُعرَّف إدارة الحالة بأنها المحافظة على قيم أو أغراض خلال حياة تطبيق الوب أو خلال مرحلة تفاعل المستخدمين مع التطبيق.

توفر ASP.NET أربعة وسائل للتوصول إلى هذا الغرض .

- حالة المستخدم (Session)
- حالة التطبيق (Application)
- الحالة المرحلية للتطبيق (Cache)
- المتحولات الساكنة

تُعرَّف إدارة الحالة بأنها المحافظة على قيم أو أغراض خلال حياة تطبيق الوب أو خلال مرحلة تفاعل المستخدمين مع التطبيق.

توفر ASP.NET أربعة وسائل للتوصول إلى هذا الغرض .

- حالة المستخدم (Session): يتم التحكم والإبقاء على حالة المستخدم باستخدام الغرض Session الذي يسمح بالإبقاء على بيانات لمدة محدودة من الزمن (عادة ما تكون 20 دقيقة) لمستخدم محدد. تكون بيانات غرض Session الخاصة بمستخدم معزولة تماماً عن بيانات هذا الغرض بالنسبة لمستخدم آخر.

على سبيل المثال يمكننا استخدام الغرض Session إذا أردنا تتبع اليافطات الإعلانية التي قمنا بإظهارها لمستخدم معين. فإذا لم يقم المستخدم بالتفاعل مع الموقع خلال الزمن المعين، ستنتهي صلاحية البيانات وسيتم حذفها لهذا المستخدم.

- حالة التطبيق (Application):يتم التحكم بحالة التطبيق والإبقاء عليها من خلال الغرض Application الذي يسمح بالحفاظ على البيانات الخاصة بتطبيق معين. تكون هذه البيانات متاحة لجميع المصادر(صفحات وب ، خدمات وب ...).
- الحالة المرحلية للتطبيق (Cache): يتم التحكم بالحالة المرحلية باستخدام الغرض Cache. يكون عمل الغرض Cache مشابه للغرض Application في كونه مشترك أي يمكن الوصول إليه من أي مصدر ضمن تطبيق الوب. إلا أن لهذا الغرض مزايا إضافية. فعلى سبيل المثال، قد يلزمنا أن نقوم بتأهيل غرض مستخدم من جميع صفحات ASP.NET من ملف XML عند بدء تشغيل تطبيق الوب' عندها نستطيع أن نخزن هذا الغرض ضمن الغرض Cache وننشأ مؤشراً لهذا الغرض على ملف XML مصدر البيانات. في حال تغير ملف البيانات ستتحسس ASP.net ذلك وستقوم بإلغاء صلاحية العنصر الذي تمت عليه التغييرات.

- المتحولات الساكنة: بالإضافة إلى استخدام الغرض Application و Cache يمكننا استخدام أحد التسهيلات التي تقدمها البيئة غرضية التوجه في ASP.NET وهي المتحولات الساكنة. حيث يمكننا التصريح عن متغيرات ساكنة وعندها سيتم إنشاء نسخة وحيدة من هذه المتغيرات مهما كان عدد نسخ الصف الذي جرى إنشاؤه.

يمكن الوصول إلى هذه المتغيرات عبر التطبيق وفي بعض الحالات تكون أكثر فائدة من الغرض Application.

استخدام أغراض الحالة

يكون استخدام الغرض Application والغرض Session في ASP.NET مطابقاً لاستخدامهما في ASP.

إذ يكفي إسناد قيمة وتحديد سلسلة محارف كمفتاح للعنصر أي من الشكل.

```
' Set an Application value
Application("SomeValue") = "my value"
' Read an Application value
Dim someString As String
someString = Application("SomeValue")
```

```
' Set a Cache value
Cache("SomeValue") = "my value"
' Read a Cache value
Dim someString As String
someString = Cache("SomeValue")
```

```
' Set a Session Cache value
SessionCache("SomeValue") = "my value"
' Read a session value
Dim someString As String
someString = session ("SomeValue")
```

سنتعرف في الشرائح القادمة على كل غرض من أغراض الحالة بالتفصيل.

يكون استخدام الغرض Application والغرض Session في ASP.NET مطابقاً لاستخدامهما في ASP.

إذ يكفي إسناد قيمة وتحديد سلسلة محارف كمفتاح للعنصر أي من الشكل.

سنتعرف في الشرائح القادمة على كل غرض من أغراض الحالة بالتفصيل.

استخدام أغراض الحالة– الغرض Session

تُعتبر نسخة الغرض Session في ASP.NET، نسخة محسنة عن سابقتها في ASP. فقد حلت النسخة المعدلة من الغرض Session مجموعة من المشاكل التي عانت منها النسخة القديمة وهي:

- دعمها لما يسمى ب (مجموعة من مخدمات الوب – farm) وهي عبارة عن وجود أكثر من مخدم وب حيث يمكن أن يجري تحويل الطلب لمخدم وب مختلف في كل مرة. كانت هذه العملية في النسخة القديمة تسبب في بعض الأحيان خللاً بسبب كون معلومات الجلسة مخزنة محلياً على المخدم .
أما في النسخة الجديدة فيمكن أن يتم إعداد المخدمات للتشارك بمخزن وحيد لمعلومات الأغراض Session.
وفي هذه الحالة لا يهم تحويل طلب المستخدم إلى مخدم آخر إذا كان هذا المخدم يستطيع الوصول إلى معلومات الجلسة.
- دعم استخدام الغرض Session حتى دون تفعيل أو دعم الكعكات من قبل المستعرض.

البرمجة باستخدام الغرض Session :

يكون الغرض Session مخصصاً لتخزين البيانات كل مستخدم ضمن تطبيقASP.NET. يتم التعامل مع الغرض Session كالغرض HashTable، ويجري تخزين البيانات على قاعدة قيمة/مفتاح.

إسناد وقراءة قيمة من غرض Session:

يمكن إسناد قيمة أو غرض إلى غرض Session باستخدام التعبير التالي في VB.NET :

```
Session("[string key]")=Object
```

و بصيغة مشابهة في #C:

```
Session [ "[string key]" ]=Object;
```

على سبيل المثال، إذا أردنا استعادة القيمة الممثلة بسلسة محارف "Hello world" المخزنة في غرض Session باستخدام المفتاح "SimpleSession"، يكفي كتابة:

```
Dim sessionValue As String
sessionValue = Session("SimpleSession")
```

و في #C:

```
string sessionValue;
sessionValue = Session["SimpleSession"];
```

أما بالنسبة للأنماط المختلفة عن سلاسل المحارف والأغراض فسنحتاج إلى تحويل القيمة حسب النمط المطلوب. فعلى سبيل المثال، إذا أردنا استعادة صف مخصص قمنا بإنشاءه وتخزينه ضمن غرض الجلسة وليكن الغرض PurchaseOrder، نكتب :

```
Dim po As PurchaseOrder
po = CType(Session("po"), PurchaseOrder)
```

أو باستخدام #C نكتب :

```
PurchaseOrder po;
po = (PurchaseOrder) Session["po"];
```

نلاحظ أننا قمنا بعملية تحويل قسري إلى النمط المحدد بالصف PurchaseOrder.

بعض خصائص الغرض Session

- يقدم الغرض Session مجموعة من الخصائص لتحديد الطريقة التي يعمل بها هذا الغرض:

- الخاصة IsCookieless تعيد القيمة True أو false لتحديد كون غرض Session يعمل باستخدام الكعكات أو بدون استخدامها .

تكون الوضعية التلقائية للعمل هي False بمعنى أن العمل يتم باستخدام الكعكات.

- الخاصة IsReadOnly تعيد إحدى القيم True أو False وتحدد كون غرض Session يعمل في وضعية القراءة فقط. هذه الوضعية مفيدة في الحالات التي لا نريد أن نسمح فيها بتغيير القيم المخزنة في الغرض Session.

تكون القيمة التلقائية لهذه الخاصة هي False أي أنها تسمح بالقراءة والتعديل على البيانات المحتواة في الغرض Session.

- الخاصة Mode والتي تحدد نمط تخزين معلومات الغرض Session. هناك مجموعة من الاحتمالات للقيم الممكنة تتضمن sqlserver، وstateServer، وoff، و inproc.

إدارة حالة التطبيق باستخدام الغرض Application

بعكس الغرض Session المخصص لتخزين معلومات عن كل مستخدم على حدة، يشكل الغرض Application مكان تخزين لمعلومات مشتركة للتطبيق.
يكون هذا المخزن المشترك مفيد جداً خصوصاً في حالة الرغبة بإنشاء مصادرمشتركة بين كل المستخدمين .

يستخدم غرض Application الغرض HashTable كآلية تطبيق إذا يخزن البيانات على شكل ثنائيات مفتاح/قيمة.

لايدعم الغرض Application مفهوم تخزين البيانات بصورة منفصلة عن إجراء ASP.NET بل يتم تخزين البيانات ضمن الإجراء، فإذا تمت عملية إعادة استخدام هذا الإجراء مرة أخرى أو إعادة إقلاعه ستزول جميع البيانات المخزنة فيه.

تكون المقايضة هنا على حساب الكسب في السرعة، فقد خسرنا الاستقلالية ولكننا حصلنا على سرعة أكبر، لأن استرجاع المعلومات باستخدام نفس الإجراء أسرع من استدعاء إجراء آخر لإتمام عملية الاسترجاع.

اسناد القيم واسترجاعها من الغرض Application:

تكون الصيغة المستخدمة للوصول إلى القيم باستعمال الغرض Application مشابهة لتلك المستخدمة مع الغرض Session باستثناء وحيد:

- باعتبار أنه يمكن الوصول إلى الغرض Application من بيئة متعددة المستخدمين لا بد من ضمان تزامن التحديثات على هذا الغرض. هذا يعني أنه لا بد –وفي كل مرة نقوم فيها بتغيير البيانات– من منع الوصول إليها لأي مستخدم أو تطبيق آخر لحين إنهاء عملية التحديث. لحسن الحظ يوفر الغرض Application آلية لتمكين هذا العمل وذلك باستخدام طرق إقفال للبيانات.

قراءة وكتابة بيانات الغرض Application

يمكننا قراءة بيانات الغرض Application من خلال استخدام صيغة مشابهة لما يلي:

في VB.NET:

```
Application (“HitCounter”)=1
```

وباستخدام #C من الشكل:

```
Application[“HitCounter”]=10;
```

وبشكل مشابه إذا أردنا قراءة القيم المخزنة في غرض Application يكفي أن نستخدم صيغة من الشكل:

في VB.NET:

```
Dim hitCount as integer
hitCount=Application(“HitCounter”)
```

في #C:

```
Int HitCount= Application[“HitCounter];
```

أما بالنسبة لطرق الإقفال المطبقة فلا بد أن نقوم بمنع المستخدمين والتطبيقات الأخرى من إجراء أي عملية تحديث أو تحرير للبيانات لحين تحرير هذا الغرض من قبل المستخدم أو التطبيق الحالي.

إذا كانت البيانات المشاركة تتغير بتكرار عالي يكون استخدام الغرض Application لحفظ الحالة على الأغلب غير أمثلي.

قراءة وكتابة بيانات الغرض Application

لنلاحظ المثال التالي الذي يبين طريقة استخدام الأقفال على الغرض Application:

في VB.NET:

```
Public Sub Application_OnStart()
Application("HitCount") = 0
End Sub
Public Sub Application_OnBeginRequest()
Application.Lock()
Application("HitCounter") = Application("HitCounter") + 1
Application.UnLock()
End Sub
```

وفي C# :

```
public void Application_OnStart() {
Application["HitCounter"] = 0;
}
public void Application_OnBeginRequest(){
Application.Lock();
int tmpInt = (int)Application["HitCounter"];
Application["HitCounter"] = tmpInt + 1;
Application.UnLock();
}
```

قمنا في النص البرمجي السابق باستدعاء الطريقة Lock() الخاصة بالغرض Application لضمان عدم محاولة أي إجراء آخر تحديث بيانات HITCounter في نفس الوقت.

عند إيقاف الإجراء ASP.NET أو إعادة استخدامه ستزول جميع البيانات كما ذكرنا. حيث يجري إطلاق حدث Application_OnEnd عند إيقاف التطبيق أو إعادة استخدامه. يمكن استخدام معالج هذا الحدث في حال الرغبة بإجراء عملية حفظ للبيانات ضمن ملف أوضمن قاعدة بيانات.

في حال عدم استدعاء الطريقة Unlock ستقوم ASP.NET تلقائياً بتحرير الغرض Application عند الانتهاء من تلبية الطلب، أو انتهاء الزمن المخصص لتلبية الطلب، أو في حال ظهور خطأ ما.

إدارة الحالة المرحلية – الغرض Cache

يستخدم العديد من المطورين التطبيق كمخزن لحالة الموارد المستخدمة بكثرة. كما هي الحال في قراءة ملف XML يحتوي دليل منتجات. إذ يجري عندها تخزين الغرض الممثل لملف XML هذا ضمن ذاكرة التطبيق.

ما الذي يحصل في حال تغير ملف XML الممثل لدليل المنتجات؟

في معظم الحالات يستخدم المطورون الغرض Application لتدارك التغيير بإعادة تشغيل التطبيق فيجري تحديث المعلومات.
يكمن الهدف من وراء تصميم الغرض Cache في منح المطورين مزايا الغرض Application إضافة إلى مزايا أخرى مثل استبعاد عنصر من الغرض cache عندما يتم تغييره حيث تصبح مسؤولية المطور إضافة العنصر من جديد إلى الغرض Cache باستخدام النص البرمجي.

يشبه الغرض Cache الصف Cache الموجود ضمن فضاء الأسماء System.Web.Cache إضافة إلى كونه يعمل كغرض HashTable على قاعدة قيمة\مفتاح كحال الأغراض Application و Session.

يدعم الغرض Cache المزايا التالية:

- انتهاء الصلاحية للبيانات بناء على اعتمادها على أغراض أخرى يمكن أن تكون مفاتيح غرض Cache أخرى ،ملفات ، التاريخ والوقت . عند أي تغيير على أي من البيانات أو الأغراض التي يعتمد عليها الغرض Cache يتم اعتبار البيانات غير صالحة وتتم إزالتها من بيانات الغرض Cache.
- إدارة الأقفال: تتم هذه العملية بشكل مشابه للذي يتم في حالة الغرض application. فبعكس الغرض Application، يمتلك الصف Cache إدارة أقفال داخلية، مما لا يجعلنا مضطرين قسرياً لاستخدام Lock() و Unlock() عند التعديل على بيانات الغرض Cache مع الأخذ بعين الاعتبار أننا ما نزال نحتاج إلى إدارة العمل المتزامن للأغراض المخزنة في الغرض Cache كما فعلنا مع الغرض Application.
- إدارة المصادر : عندما يكتشف الغرض Cache ضغط على الذاكرة يقوم بالمرور على بياناته ويُبعد العناصر الأقل استخداماً. لذا وقبل طلب أي عنصر يتوجب علينا التحقق من وجوده أولاً.
- الاستدعاء الراجع: يوفر الغرض Cache إمكانية السماح بتشغيل نص برمجي عند إزالة عنصر من هذا الغرض.

إدراج العناصر في الغرض Cache

يدعم الغرض Cache طريقتين لإدراج العناصر:

الطريقة الضمنية: هذه الطريقة مألوفة بالنسبة لنا كونها نفسها المستخدمة مع الأغراض Session و Application باستخدام الأزواج مفتاح\قيمة وتكون الصيغة من الشكل:

في VB.NET:

```
Dim productDataSet As New DataSet()
' Populate DataSet
Cache("products") = productDataSet
```

في #C:

```
DataSet productDataSet = new DataSet();
// Populate DataSet
Cache["products"] = productDataSet;
```

الطريقة الصريحة : باستخدام الطريقة ()Insert. تسمح هذه الطريقة بتحديد علاقات خاصة كعلاقات الاعتماد كما في الصيغة:

في VB.NET:

```
Dim productDataSet As New DataSet()
' Populate DataSet
Cache.Insert("products", productDataSet, Nothing)
```

في #C:

```
DataSet productDataSet = new DataSet();
// Populate DataSet
Cache.Insert("products", productDataSet, null)
```

عند استخدام الغرض Cache سنستخدم على الأغلب الطريقة الصريحة باستعمال ()Insert.

إنتهاء الصلاحية المبني على الاعتمادية

يعد انتهاء الصلاحية المبني على الاعتمادية آلية في غاية القوة إذ أنه يمكننا من إنشاء علاقة بين عنصر من الغرض Cache وملف، لفترة محددة من الزمن. مثال:

```
<%@ Import Namespace="System.Xml" %>
<%@ Import Namespace="System.Xml.Xsl" %>
<Script runat="server">
Public Sub Page_Load(sender As Object, e As EventArgs)
Dim dom As XmlDocument
Dim xsl As New XslTransform()
' Do we have the Wrox Pro ASP.NET 2nd Ed book in the Cache?
If (IsNothing(Cache("1861007035.xml"))) Then
CacheStatus.Text = "Item not present, updating the Cache..."
UpdateCache("1861007035.xml")
```

```
Else
CacheStatus.Text = "Retrieving from Cache"
End If
' Load the transform
xsl.Load(Server.MapPath("book.xsl"))
dom = CType(Cache("1861007035.xml"), XmlDocument)
BookDisplay.Document = dom
BookDisplay.Transform = xsl
End Sub

Public Sub UpdateCache(strItem As String)
Dim strPath As String
Dim dom As New XmlDocument()
' Determine the file path of the file to monitor
strPath = Server.MapPath(strItem)
' Load the file into an Xml Dom
dom.Load(strPath)
' Create a CacheDependency on the file
Dim dependency as New CacheDependency(strPath)
' Cache the XML document
Cache.Insert(strItem, dom, dependency)
End Sub
</Script>
Status: <asp:label id="CacheStatus" runat=server/>
<br>
<asp:xml id="BookDisplay" runat=server/>
```

تظهر في المثال السابق التعليمات المتعلقة بالغرض Cache بلون مغاير، سنهمل الآن التعليمات الأخرى المتعلقة بـــ XML كوننا سنفرد جلسة خاصة لهذا الموضوع، فما يهمنا الآن هو ملاحظة كيف يتم التأكد من وجود العنصر ضمن الغرض Cache و كيف يتم استخدامه بعد تحويله إلى صيغة وثيقة XML باستخدام CType. أو كيف يمكن استخدام الإجرائية `UpdateCache` لإسناد غرض وثيقة XML الحامل للملف من جديد إلى غرض Cache باستخدام التعليمة Insert() بعد تحديد غرض CacheDpendency جديد لتأسيس علاقة بين الملف وغرض Cache.

يعد انتهاء الصلاحية المني على الاعتمادية آلية في غاية القوة إذ أنه يمكن من إنشاء علاقة بين عنصر من وملف، لحظة محددة من الزمن ، أو مفتاح آخر ضمن نفس الغرض. الغرضCache

مثال

وضح المثال في الشريحة السابقة كيفية مراقبة احتواء الغرض Cache على غرض ملف XML. ولكن ماذا لو أردنا إزالة عنصر أو أكثر من الغرض Cache عند تعديل عنصر آخر ضمن هذا الغرض.

يتم تأمين هذا الخيار عبر ما يسمى الاعتمادية على أساس المفتاح. تكون الصيغة المستخدمة لدعم هذا الخيار مشابهة لتلك المستخدمة عند استخدام الاعتمادية على أساس الملف. فعلى سبيل المثال نستطيع بسهولة تغيير النص البرمجي للمثال السابق بحيث يدعم الاعتمادية على أساس المفتاح بتعديل مجموعة من الأسطر.

```
<%@ Import Namespace="System.Xml" %>
<Script runat="server">
Public Sub Create(sender As Object, e As EventArgs)
' Create the Cache entry for the dependency relationship
' the value of the key doesn't matter
Cache("booksDependencyKey") = "Book Dependency"
' Create a string array with the key names for the
' dependencies to be created upon
Dim dependencyKey(0) As String
dependencyKey(0) = "booksDependencyKey"
' Create a CacheDependency on this key
Dim dependency as New CacheDependency(nothing, dependencyKey)
' Cache the XML document
Cache.Insert("1861007035.xml", Load("1861007035.xml"), dependency)
Status()
End Sub
Private Function Load(xmlFile As String) As XmlDocument
Dim dom As New XmlDocument()
dom.Load(Server.MapPath(xmlFile))
Return dom
End Function
Public Sub Invalidate(sender As Object, e As EventArgs)
Cache.Remove("booksDependencyKey")
Status()
End Sub
Public Sub Status()
If (IsNothing(Cache("1861007035.xml"))) Then
lblStatus1.Text = "No value..."
Else
lblStatus1.Text = "Cache entry exists..."
End If
End Sub
</Script>
<form runat=server>
<input type="submit" OnServerClick="Create"
value="Create Cache Entries" runat="server" />
<input type="submit" OnServerClick="Invalidate"
value="Invalidate Key" runat="server" />
</form>
Status for cache key: 1861007035.xml: <b><asp:label id="lblStatus1"
runat=server/></b>
```

عند تشغيل النص البرمجي السابق نحتاج إلى ضغط زر Create Cache Entries لإنشاء بيانات الغرض Cache الخاص بملف الــXML وبعلاقة الاعتماد.

بعدها يمكننا نقر Invalidate Key الذي يقوم بإطلاق الحدث Invalidate.

نقوم ضمن هذا الحدث بإزالة المفتاح BooksDependencyKey بصورة صريحة.

يقودنا هذا وبصورة قسرية إلى إزالة بيانات الغرض Cache لوثيقة XMl أيضاً.

بالإضافة إلى الاعتمادية على أساس الملف والاعتمادية على أساس المفتاح يمكننا إنشاء إعتمادية على أساس القيمة الزمنية.

مثال:

إذا أردنا تخزين جميع عناوين الكتب في جدول واحد من قاعدة البيانات، وعلمنا أنه يجري تحديث هذه البيانات مرة في الأسبوع، يمكننا أن ندرج الغرض Dataset الممثل لهذه البيانات ضمن الغرض Cache بصورة صريحة مع صلاحية تنتهي بعد 60 دقيقة.

سيوفر علينا هذا الأمر الاتصال بقاعدة البيانات عند كل طلب مع ضمان أنه حتى في حال ظهور تحديث على البيانات فإن البيانات التي تتم استعادتها ستكون محدثة خلال الساعة التالية.

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<script runat=server>
Private DSN As String
Public Sub Page_Load(sender As Object, e As EventArgs)
Dim strCacheKey As String
Dim titlesDataSet As DataSet
strCacheKey = "Titles"
If (IsNothing(Cache(strCacheKey))) Then
lblStatus.Text = "Getting data from database..."
LoadTitles(strCacheKey)
Else
lblStatus.Text = "Getting data from Cache..."
End If
titlesDataSet = CType(Cache(strCacheKey), DataSet)
TitleList.DataSource = titlesDataSet
TitleList.DataBind()
```

```
End Sub
Public Sub LoadTitles(strCacheKey As String)
Dim connection As SqlConnection
Dim command As SqlDataAdapter
Dim sqlSelect As String
Dim strDsn As String
Dim dataset As New DataSet()
sqlSelect = "Select title, pub_id, price, notes, pubdate FROM
titles"
strDsn = "server=localhost;uid=sa;pwd=;database=pubs"
connection = New SqlConnection(strDsn)
command = New SqlDataAdapter(sqlSelect, connection)
command.Fill(dataset, "Author-Titles")
Cache.Insert(strCacheKey, dataset, nothing, _
DateTime.Now.AddMinutes(60), TimeSpan.Zero)
End Sub
</script>
<font size=6>
<asp:label id="lblStatus" runat="server"/>
</font>
<P>
<ASP:DataGrid id="TitleList" HeaderStyle-BackColor="#aaaadd"
BackColor="#ccccff"
runat="server" />
```

أحداث التطبيق

تجعل أحداث التطبيق في ASP.NET الكثير من الأعمال أسهل.

توفر الأحداث طرق جيدة للتحكم بتنفيذ النص البرمجي وتنظيمه.

نستطيع استخدام أحداث التطبيق بإحدى طريقتين:

- **استخدام النموذج الأولي في ملف Global.asax :**

سنقوم في هذه الحالة ببساطة بإضافة ملف النموذج الأولي للأحداث. تشبه هذه الطريقة لما ما كان يجري في Asp سلف Asp.Net حيث كانت تجري كتابة معالجات لأحداث مثل Application_OnStart أو Session_OnEnd.

- **تحرير وحدات Http النمطية:**

تعطينا وحدة Http النمطية المجال للعمل على طلب Http قبل تخديمه من قبل ASP.NET
نستطيع استخدام هذه الوحدات النمطية لتحرير حلول مخصصة لتطبيق ASP.Net معين.

ملاحظة :

- يمكن أن يكون هناك استجابة من Global.asax ووحدة Http النمطية على نفس الحدث.
- تدعم ASP.NET، 18 حدث خاص بالتطبيق كما تسمح لنا بإضافة أحداثنا المخصصة عن الحاجة.

صيغة أحداث التطبيق والنموذج الأولي

تأخذ أحداث التطبيق في Global.asax الصيغة :

في VB.NET

```
Public Sub Application_OnStart(sender As Object, e As EventArgs)
End Sub
```

أما في C#:

```
public void Application_OnStart(Object sender, EventArgs e) {
}
```

يحدد المعامل التي قمنا باستخدامه الغرض الذي قام بإطلاق الحدث إضافة إلى المعامل EventArgs الذي يتيح المجال للغرض الذي أطلق الحدث بتزويدنا بمعلومات وتفاصيل عن هذا الحدث.

يمكننا استخدام الصيغة السريعة دون تحديد اسماء المعاملات:

```
Public Sub Application_OnStart()
End Sub
```

وفي C#:

```
public void Application_OnStart() {
}
```

لا يمكننا في الصيغة السابقة الوصول إلى EventArgs أو Sender لذلك يفضل استخدام الصيغة المفصلة لأنها تمكننا من تحكم أفضل.

يمكننا تصنيف أحداث التطبيق إلى تصنيفين أساسيين:

- أحداث يتم إطلاقها عند كل طلب؛
- أحداث شرطية يتم إطلاقها عند تحقق شرط مثل الأحداث التي تم إطلاقها عند حدوث خطأ ما.

صيغة أحداث التطبيق والنموذج الأولي

الأحداث السابقة للطلب:

تُعرَّف أحداث التطبيق السابقة للطلب بأنها الأحداث التي يتم إطلاقها أثناء كل طلب يتم توجيهه إلى تطبيق ASP.NET مثل أحداث بداية ونهاية الطلب:

- حدث Application_OnBeginRequest : يجري إطلاق هذا الحدث مع كل طلب تتم معالجته من قبل ASP.NET. يختلف هذا الحدث عن الحدث المألوف Application_OnStart في نسخة ASP الذي يتم إطلاقه مرة واحدة عند تشغيل التطبيق.
يمكننا استخدام هذا الحدث لتنفيذ نص برمجي قبل أي صفحة أو خدمة وب أو قبل أن يتمكن أي معالج طلب HTTP من بدء العمل على أي طلب.

- الحدث Application_OnAuthenticateRequest : يتم إطلاق هذا الحدث عندما تصبح ASP.NET جاهزة للتحقق من الطلب. يُسهّل هذا الحدث علينا بناء نظام تحقق مخصص بحيث نستطيع فحص الطلب وتنفيذ النص البرمجي الذي يحدد فيما إذا كان الطلب مقبولاً أم لا.

- الحدث Application_OnAuthorizationRequest : يعمل هذا الحدث بصورة شبيهة لحدث التحقق حيث يجري إطلاق هذا الحدث عندما يصبح الطلب جاهزاً ليتم إعطاءه السماحيات على مصدر ما. يُمَكننا هذا الحدث من إنشاء وتنفيذ نصوص برمجية تحدد السماحيات الخاصة بالطلبات.

- الحدث Application_OnResolveRequestCache : تمتلك ASP.NET آليات تخزين صفحات وخدمات الوب. ف بدل تنفيذ الصفحة لكل طلب يمكن أن يجري تنفيذ الصفحة لمرة وتخديم طلبات أخرى

بالنسخة الستاتيكية الناتجة.

ينطلق هذاالحدث عندما تصبح ASP.NET جاهزة لتحديد كون الطلب سيتم تخديمه من الخبئ أم لا.

صيغة أحداث التطبيق والنموذج الأولي

- الحدث Application_OnAcquireRequestState:

يتم إطلاق هذا الحدث عندما تصبح ASP.NET جاهزة لتلقي بيانات الجلسة من داخل إجراء ASP.NET أو من خارجه أو من SQLServer.

إذا قررنا استخدام غرض Session مخصص كغرض XMLSession مثلاً، يمكننا تأهيل قيم ذلك الغرض باستخدام هذا الحدث.

عند تسليم الطلب للصفحة أو لخدمة الوب سيكون الغرض XmlSession مؤهلاً بالقيم.

- الحدث Application_OnPreRequestHandlerExecute:

يتم إطلاق هذا الحدث قبل استدعاء معالج تخديم الطلب. في معظم الحالات يكون هذا المعالج هو معالج Page.

بعد إطلاق الحدث Application_OnPreRequestHandlerExecute يتسلم معالج HTTP الطلب.

يتم إطلاق الحدث التالي بعد الانتهاء من معالجة الطلب.

- الحدث Application_OnPostRequestHandlerExecute : يتم إطلاق هذا الحدث فور الانتهاء من معالجة الطلب. في هذه المرحلة يكون لدى الغرض Response بيانات يجري إرسالها إلى الزبون.

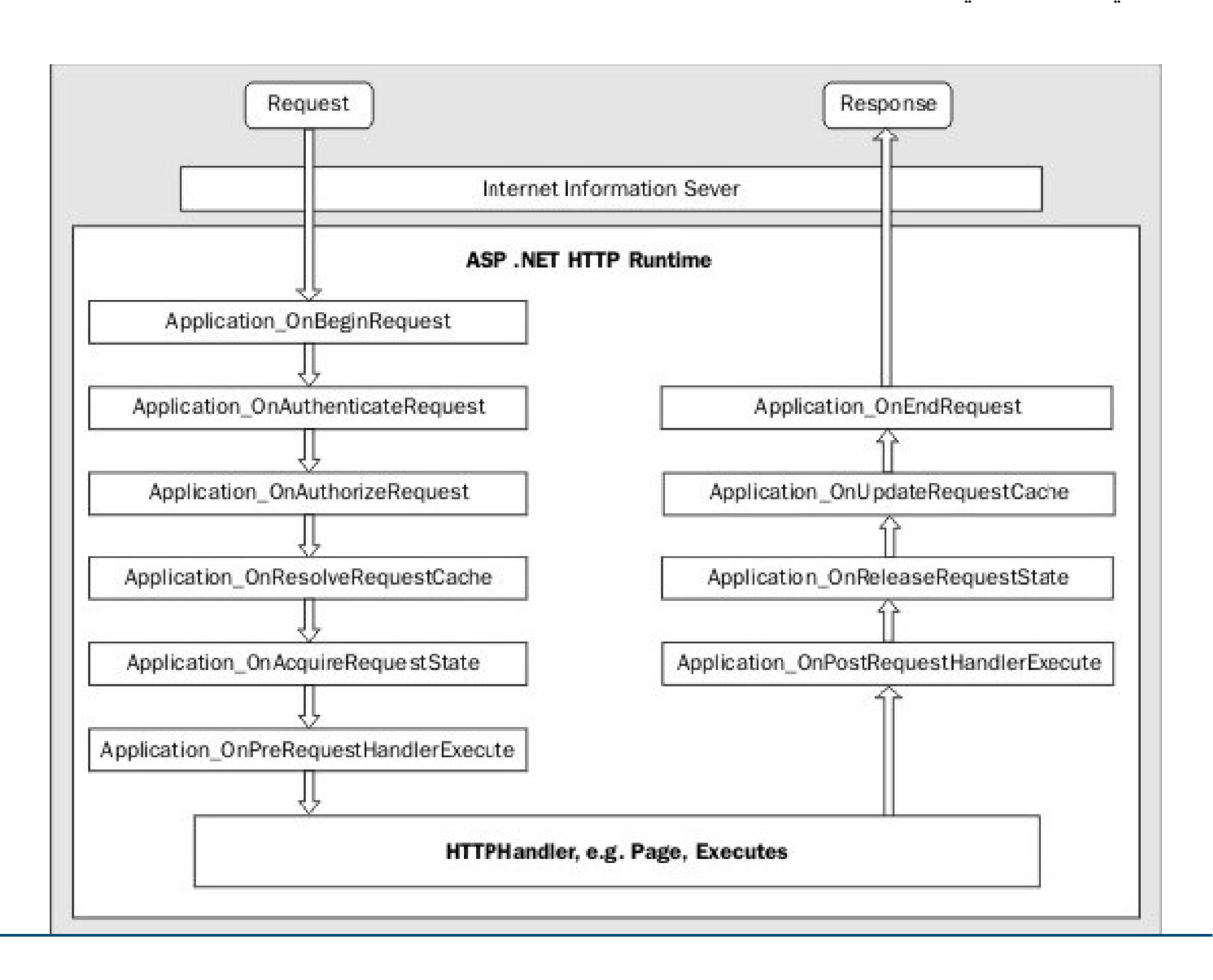
- الحدث Application_OnReleaseRequestState يقوم هذا الحدث بتحرير بيانات الجلسة وتحديث المعلومات المخزنة عند اللزوم. بعد إطلاق هذا الحدث لا يمكن تحديث معلومات الجلسة.

- الحدث Application_On_UpdateRequestCache يتم إطلاق هذا الحدث عندما تقوم ASP.NET بتحديث خرج الذاكرة الخبيئة بالطلب الحالي .

- الحدث Application_OnEndRequest: هذا هو الحدث الأخير الذي سيتم إطلاقه ويسمح لنا التحكم باستجابة التطبيق قبل إرسال ترويسات HTTP مع الاستجابة.

صيغة أحداث التطبيق والنموذج الأولي

فيما يلي مخطط تدفقي للأحداث الخاصة بالتطبيق:



كما هو مبين أعلاه يقوم IIS بتلقي الطلب وتسليمه إلى الإجراء ASP.NET، عندها تبدأ الأحداث المذكورة سابقاً بالانطلاق ابتداءاً من Application_OnBeginRequest وبصورة فورية.
قبل أن يتم استدعاء معالج HTTP يجري إطلاق Application_OnPreRequestHandlerExecute.
وفور انتهاء تنفيذ معالج HTTP من التنفيذ ينطلق الحدث
.Application_OnPostRequestHandlerExecute
أخيراً وقبل تسليم الاستجابة إلى IIS يتم إطلاق الحدث Application_OnEndRequest.

صيغة أحداث التطبيق والنموذج الأولي

هناك حدثان آخران بنتميان إلى مجموعة الأحداث التي تتكرر مع كل طلب ولكنهما ينطلقان عندما تصبح البيانات جاهزة ليتم إرسالها إلى الزبون.

تفعّل ASP.NET بشكل تلقائي Buffer مما يعني أن المخدم لن يبدأ بإرسال البيانات إلى طالبها إلا حين تصبح هذه البيانات جاهزة.

عند تفعيل Buffering، ينطلق الحدثان التاليان بعد Application_OnEndRequest:
- الحدث Application_OnPreSendRequestHeaders قبل إرسال ترويسات HTTP إلى الزبون صاحب الطلب.
- الحدث Application_OnPreSendRequestContent قبل إرسال الاستجابة إلى الزبون مرسل الطلب.

الأحداث الشرطية الخاصة بالتطبيق

ذكرنا وجود تصنيفين لأحداث التطبيق في ASP.NET: تناولنا الأول المتعلق بالأحداث التي تتكرر مع كل طلب وسنغطي الآن الأحداث الشرطية.

تُعرَّف الأحداث الشرطية بأنها الأحداث التي يمكن أن تظهر أو لا تظهر خلال معالجة طلب.
فعلى سبيل المثال، عند تشغيل التطبيق للمرة الأولى ينطلق الحدث Application_OnStrat أو عند ظهور خطأ في التطبيق ينطلق الحدثِ Application_Error .

• الحدث Application_OnStart: ينطلق هذا الحدث عندما يتم تشغيل التطبيق للمرة الأولى وهو بعكس الحدث Application_OnBeginRequest لا يُطلق مع كل طلب. يمكننا استخدام هذا الحدث لتجهيز تطبيقنا لتخديم الطلبات.

مثال: استخدام هذا الحدث لتأسيس اتصال مع قاعدة البيانات والحصول على بعض البيانات.

• الحدث Application_OnEnd: يظهر هذا الحدث لمرة واحدة أيضاً وهو مقابل للحدث Application_OnStart ويتم إطلاقه عندما يتم إيقاف تشغيل التطبيق. يمكن استخدام هذا الحدث كحدث تنظيف. يتضمن عمليات إغلاق الاتصال مع قاعدة البيانات تفريغ البيانات من الخبئ . في الواقع تكون معظم العمليات السابقة غير ضرورية كون CLR تقوم بعملية التنظيف تلك وتحرير الذاكرة المحجوزة للتطبيق، إلا أنه يُفضل دائماً إجراء تلك العملية بأنفسنا.

أحداث التطبيق الشرطية

• الحدث Session_OnStart يتم إطلاق هذا الحدث عندما تبدأ جلسة مستخدم ما في تطبيق ASP.NET. يمكننا استخدام هذا التطبيق لتنفيذ نص برمجي خاص بالمستخدم كعملية إسناد قيم لغرض Session.

• الحدث Session_OnEnd: يتم إطلاق هذا الحدث (المقابل لحدث بدء جلسة) عند انتهاء جلسة مستخدم. يمكن استخدام هذا الحدث لحفظ معلومات الجلسة أو للتجوال في بيانات الجلسة لتسجيل ما هو مهم لنا ضمن قاعدة البيانات .

• الحدث Application_Error: يتم إطلاق هذا الحدث عند ظهور خطأ ضمن التطبيق. يعتبر هذا الحدث شديد الأهمية لأنه يساعدنا في التقاط الأخطاء ومعالجتها وفي إرسال بريد لمدير النظام بتفصيلات عن الخطأ.

• الحدث Application_OnDisposed: يتم إطلاق هذا الحدث عندما يتم إيقاف تشغيل وينتهي CLR من تحرير ذاكرة التطبيق. ولكن هذا الحدث قليل الاستخدام فعلياً.

لايعني وجود 18 حدث في ASP.NET أنه يتوجب استخدامها كلها في كل تطبيق ويبقى تقدير استخدام الأنسب عائداً للمطور.

أمثلة

فيما يلي سنستعرض مجموعة من الأمثلة التي تتناول أحداث التطبيقات في ASP.NET

مثال إضافة تذييل إلى جميع الصفحات:

ذكرنا أن الحدث Application_OnEndRequest يتم إطلاقه في نهاية الطلب فوراً قبل إرسال الاستجابة إلى طالبها. لنفرض أن مزود خدمة أنترنت يريد إضافة تذييل إلى جميع الصفحات التي يتم تخديمها باستخدام ASP.NET. يمكن أن يكون ملف Global.asax عندها من الشكل:

```
<Script runat="server">
Public Sub Application_OnEndRequest(sender As Object, e As
EventArgs)
Response.Write("<hr size=1>")
Response.Write("<font face=arial size=2>This page was " _
& "served by ASP.NET</font>")
End Sub
</Script>
```

في المثال أعلاه استخدمنا التعبير Response.write لإظهار العبارة "This page was …"

مثال تحميل بيانات مخصصة بالمستخدم:

تقوم العديد من المواقع بتخصيص نفسها حسب طبيعة مرسل الطلب. فماذا لو أردنا التخصيص بالنسبة لمجموعة وليس لمستخدم فريد.

مثلاً إذا أردنا تقسيم المستحدمين إلى مجموعات هي Gold، وSilver، وBronze ونريد لكل مجموعة من هؤلاء الوصول إلى موارد غير متاحة للمجموعات الباقية.

نستطيع بسهولة بناء مدير حالة يساعدنا عند استخدام الحدث Application_OnAquireRequest من تحديد لمن يعود كل طلب واستحضار البيانات المناسبة له. مثال:

```
<%@ Import Namespace="System.Xml" %>
<Script runat="server">
Public Sub Application_OnAcquireRequestState( _
sender As Object, e As EventArgs)
Dim dom As New XmlDocument()
Dim customerType As String
' Grab the customerType from the QueryString
customerType = Request.QueryString("customerType")
' Check for values
If (IsNothing(customerType)) Then
customerType = "Bronze"
End If
' Load the appropriate XML file
```

```
Select Case customerType
Case "Gold"
dom.Load(Server.MapPath("Gold.xml"))
Case "Silver"
dom.Load(Server.MapPath("Silver.xml"))
Case Else
dom.Load(Server.MapPath("Bronze.xml"))
End Select
Session("WelcomeMsg") = _
dom.SelectSingleNode("/customer/welcome").InnerText
End Sub
</Script>
```

معالجة أخطاء التطبيقات

لما كانت Asp.Net تستعمل CLR يمكننا استخدام CLR لبناء تطبيقات وب.

تعتبر معالجة الاستثناءات بطريقة Try/catch أحد المزايا الأساسية في CLR. لكن وجود هذه الأمكانيات القوية التي توفرها هذه البنية، لايمنع من بناء نصوص برمجية تحوي الكثير من العلل.

فعلى سبيل المثال، قد نكتب نصاً برمجياً يقوم بالاتصال بقاعدة بيانات وبالقراءة منها. كما يمكن أن نغلف هذه النص البرمجي في كتلة try/catch، عندها يمكننا في حال عدم التمكن من الاتصال معالجة هذا الخطأ بصورة مناسبة.
ولكن ما الذي سيحصل إذا ظهر الاستثناء خارج كتلة try/catch ؟ سيتم إظهاره كخطأ تشغيل يزودنا بتفاصيل عن الخطأ، وعن مكانه وعن العمل الذي كان التطبيق يقوم به.

يمكننا بالنسبة لصفحات ASP.NET اعتماد الحدث Page_Error لالتقاط جميع الاخطاء التي لم تتم معالجتها.

أما إذا قررنا التقاط جميع الأخطاء على مستوى التطبيق فإن Application_Error يساعدنا في ذلك:

```
<%@ Import Namespace="System.Diagnostics" %>
```

```
<script language="VB" runat=server>
Public Sub Application_Error(Sender as Object, E as EventArgs)
Dim LogName As String = "Web_Errors"
Dim Message As String
Message = "Url: " & Request.Path
Message = Message + " Error: " & Server.GetLastError.ToString
' Create event log if it doesn't exist
If (Not EventLog.SourceExists(LogName)) Then
EventLog.CreateEventSource(LogName, LogName)
End if
' Fire off to event log
Dim Log as New EventLog
Log.Source = LogName
Log.WriteEntry(Message, EventLogEntryType.Error)
End Sub
</script>
```

قمنا في المثال أعلاه باستيراد فضاء الأسماء System.Diagnostics لأننا سنقوم باستخدام صفوف في هذا الفضاء لكتابة سجل الأحداث.

قمنا بعدها باستخدام الحدث Application_OnError لإنشاء ومجموعة من المتحولات.

ثم قمنا بالتأكد من وجود سجل للأحداث باستخدام الطريقة ()SourceExist. فإذا لم نجده نعمل على إنشاءه.

قمنا بعدها بإنشاء مثيل من الصف EventLog باسم Log وقمنا باستخدام الطريقة ()WriteEntry لإدراج رسالة ضمن سجل أحداث Windows.

في حال ظهور أي خطأ في تطبيقنا سيتم تسجيله ضمن سجل الأحداث المخصص المسمى Web_Errord.

لا بد من الملاحظة هنا بأننا كتبنا فقط 10 أسطر لأداء عملية قد تتطلب 60 سطراً في ASP سلف ASP.NET.

الفصل الحادي عشر، الثاني عشر، والثالث عشر

عنوان الموضوع:

إدارة البيانات في NET.

الكلمات المفتاحية:

اتصال، أمر، غرض، واصفة، تأشيرة، طريقة، خاصة.

ملخص:

تم تزويد NET. بمجموعة قوية من التقنيات التي تسمح بإدارة البيانات والتعامل مع عدد كبير من مصادر البيانات سواء كانت هذه المصادر متمثلة بقواعد بيانات علائقية أو وثائق مثل وثائق XML.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- أغراض إدارة البيانات في .NET على كيفية استخدامها

إدارة البيانات في ASP.NET

سنغطي في هذه الجلسة والجلسة القادمة المواضيع المتعلقة بإدارة البيانات في ASP.NET. ونقصد هنا بتعبير إدارة البيانات عمليات الوصول إلى البيانات المخزنة في ملفات وتطبيقات أخرى ومعالجتها.

نسمي في الإطار العام، مصادر المعلومات **بمخازن البيانات.**

يتضمن إطار عمل .NET مجموعة من الصفوف التي تتبنى تقنيات وصول متقدمة إلى البيانات المصممة خصيصاً للاستخدام مع NET.

مخازن البيانات والوصول إليها:

يرتبط مفهوم إدارة البيانات بمصادر البيانات العلائقية مثل قواعد البيانات، ولكن تقنيات إدارة البيانات في NET. تقدم إمكانيات أخرى متميزة منها: الاتصال والتعامل مع ملفات XML والتقنيات المرتبطة بها.

تاريخياً، كانت قواعد البيانات عادة مبنية على ملف وتستخدم طول سجل محدد كما هي الحال في ملفات txt.

إذ كانت تجري قراءة الملفات إلى جداول من قبل برامج قواعد البيانات أو تقنيات الوصول إلى البيانات، وكانت تُطبق قواعد موجودة في ملفات أخرى لربط سجلات من جداول مختلفة بعضها ببعض. بعد نضوج هذه التقنيات ظهرت قواعد البيانات العلائقية لتوفر طرق تخزين أفضل مع الطول الديناميكي للسجل وتقنيات وصول أكثر فعالية للبيانات. على أي حال بقي مكان التخزين الأساسي هو قاعدة البيانات .

الانتقال إلى البيئة الموزعة:

في السنوات الأخيرة، تغيرت العديد من المتطلبات وآليات عمل التطبيقات في معظم الأعمال وتم الابتعاد عن مفهوم قاعدة البيانات العلائقية المركزية، وأصبحت البيانات موزعة بين مخدمات البريد الالكتروني ووثائق المكتب وأماكن ووسائط أخرى، ضمن قواعد البيانات أيضاً.

إدارة البيانات في NET.

توجد تقنيات جديدة للوصول إلى البيانات، تتناسب مع البيئة الجديدة الموزعة التي تكلمنا عنها. لذا سنستعرض ما تقدمه NET. فعلياً في هذا المجال.

سنبدأ بإعطاء لمحة عامة عن جميع صفوف إدارة البيانات فيNET. لنرى كيف تنسجم جميع أغراض إدارة البيانات مع بيئة البرمجة المهيكلة التي تقدمها NET.

فضاء الأسماء:

تُبنى جميع صفوف إدارة البيانات العلائقية على فضاء الأسماء system.data ويطلق عادة يطلق ado.net على فضاءات الأسماء الوجودة في الجدول التالي:

فضاء الأسماء	الوصف
System.data	يتضمن جميع الأغراض الأساسية المستخدمة للوصول إلى، وتخزين البيانات في قواعد البيانات العلائقية. من هذه

الأغراض: Dataset و DataTable و DataRelation. تكون كل من هذه الأغراض مستقلة عن نمط مصدر البيانات والطريقة التي نتصل بها بهذا المصدر.	
تحتوي الصفوف الأساسية المستخدمة من الأغراض الأخرى وخاصة الأغراض العامة من فضاء الأسماء OleDb و SqlClient. بصورة عامة لا نقوم باستيراد فضاء الأسماء هذا في تطبيقاتنا.	System.data.common
يحتوي الأغراض التي تُستخدم للاتصال مع مصدر البيانات باستخدام مزود Ole-Db مثل OleDbConnection ، OleDbCommand. ترث هذه الأغراض طرق وخصائص من الصفوف المشتركة	System.data.OleDb
تحتوي الأغراض التي يمكننا استخدامها للاتصال مع مصادر البيانات عبر سياق من البيانات الجدولية الخاصة ب SQL Server فقط. حيث توفر أداء أفضل بإزالتها بعض الطبقات الوسيطة المطلوبة من اتصال OLE_DB. ترث الأغراض مثل SQLConnection و SQLCommand من الصفوف المشتركة كـــ OleDb الخصائص والطرق والأحداث.	System.data.SqlClient
تحتوي الصفوف اللازمة لاستخدام أنماط البيانات في قواعد البيانات العلائقية مثل SQLServerوالمختلفة عن تلك القياسية في NET. كأغراض SqlDateTime .SqlMoney و SqlBinary. يحسن استخدام هذه الأغراض الأداء بشكل ملحوظ ويقلل أخطاء التحويل بين أنماط البيانات.	System.Data.SqlType

توجد تقنيات جديدة للوصول إلى البيانات، تتناسب مع البيئة الجديدة الموزعة التي تكلمنا عنها. لذا سنستعرض ما تقدمه NET. فعلياً في هذا المجال.

سنبدأ بإعطاء لمحة عامة عن جميع صفوف إدارة البيانات فيNET. لنرى كيف تنسجم جميع أغراض إدارة البيانات مع بيئة البرمجة المهيكلة التي تقدمها NET.

هناك أيضاً سلسلة من فضاءات الأسماء الحاوية على صفوف يمكن استخدامها للتعامل مع ملفات XML بدلاً من استخدام قواعد البيانات العلائقية، تكون هذه الأسماء مبنية على System.Xml.

System.Xml	تحتوي الأغراض الأساسية اللازمة لإنشاء، وقراءة ، وتخزين، وكتابة ومعالجة ثائق XML بحسب توصيات W3C. تحتوي XmlDocument بالإضافة إلى سلسلة من الأغراض التي تمثل أنواع مختلفة من العقد في وثيقة XML.
System.Xml.Schema	تحتوي الأغراض المسؤولة عن إنشاء، وتخزين، ومعالجة، الهيكل والعقد المحتواة في هيكل وثيقة XML.
System.Xml.Sirialization	يحتوي مجموعة من الأغراض التي يمكن استخدامها لتحويل وثيقة XML إلى تنسيقات أخرى مختلفة مثل SOAP للنقل عبر الشبكة مثلاً.
System.Xml.Xpath	يحتوي الصفوف اللازمة لتطبيق عمليات القراءة، والتخزين والكتابة واستعلام عن وثائق XML باستخدام غرض مبني على XPath. تتضمن أغراض مثل XpathDocument و XpathNavigator والأغراض التي تمثل تعبيرات XPath.
System.Xml.Xsl	يحتوي الأغراض اللازمة لعملية تحويل ملف XML إلى تنسيقات أخرى باستخدام XSL و XSLT. يكون الغرض الأساسي فيه هو XslTransform

استيراد فضاءات الأسماء اللازمة للعمل مع مصادر البيانات

لا بد للصفحات التي تستخدم أغراض من مكتبة صفوف إطار العمل NET. أن تستورد فضاءات الأسماء

الحاوية على الإغراض التي تريد إنشاء مثيل عنها. يجري استيراد الكثير من هذه الفضاءات تلقائياً.
لكن فضاء أسماء إدارة البيانات ليس من تللك الفضاءات التي يتم استيرادها تلقائياً، لذا يجب علينا استيراده بصورة صريحة في النص البرمجي:

استيراد فضاء الأسماء System.Data:

للوصول إلى قواعد البيانات العلائقية لا بد لنا من استخدام فضاء الأسماء System.Data على الأقل وأيٍ من System.Data.OleDb أو System.Data.SqlClient اعتماداً على الطريقة التي نود الاتصال بها مع مصدر البيانات وذلك بالصيغة:

```
<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.OleDb" %>
```

أو

```
<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.SqlClient" %>
```

يمكن في VB.NET استخدام Imports وفي C# استخدام Using

هناك حالات خاصة نضطر فيها لاستيراد فضاءات أسماء أخرى كحالة إنشاء عنصر من الغرض DataTableMapping، حيث يتوجب علينا استيراد فضاء الأسماء System.Data.Common. كما يتوجب علينا استيراد System.Data.SqlType عندما نريد استخدام أنماط بيانات من الأنواع التي يستخدمه SqlServer

استيراد فضاء الأسماء System.Xml:

للوصول إلى بيانات XML باستخدام الأغراض في صفوف مكتبات Net.، يمكننا غالباً غض النظر عن استيراد فضاء الأسماء الأساسي System.Xml .
على كل حال سنضطر إلى استيراد فضاء الأسماء System.Xml.Xpath عندما نريد إنشاء غرض XpathDocument.
وسنضطر أيضاً لاستيراد فضاء الأسماء System.Xml.Xsl عند استخدام الغرض XslTransform عند إجراء عمليات تحويل من جهة المخدم على وثائق XML.
كما يلزمنا استيراد فضاء أسماء System.Xml.Schema عند العمل مع الــ Schemas (المخططات والهياكل).
أما بالنسبة لأغراض مثل XmlValidatingReader فلا نحتاج فيها إلى استيراد System.Xml.Schema.

في حال نسينا استيراد أي من فضاءات الأسماء اللازمة سنحصل على رسالة خطأ من الشكل:

Server Error in '/7035' Application.

Compilation Error

Description: An error occurred during the compilation of a resource required to service this request. Please review the following specific error details and modify your source code appropriately.

Compiler Error Message: BC30002: Type 'SqlConnection' is not defined.

Source Error:

```
Line 43:
Line 44:        'create a new Connection object using the connection string
Line 45:        Dim objConnect As New SqlConnection(strConnect)
Line 46:
Line 47:        'open the connection to the database
```

Source File: C:\Inetpub\wwwroot\7035\data-access\data01\datareader-sql.aspx **Line:** 45

Show Detailed Compiler Output:

Show Complete Compilation Source:

Version Information: Microsoft .NET Framework Version:1.0.3617.0; ASP.NET Version:1.0.3617.0

أغراض ADO.NET الأساسية

اعتمدت طريقة الوصول التقليدية إلى البيانات –والتي استخدمت تقنية ADO– على غرض رئيسي وحيد هو Recordset، وكانت التقنية تتلخص في:

- تأسيس اتصال بقاعدة البيانات باستخدام مزود OLE-DB أو ODBC عبر OLE-DB
- تنفيذ أوامر على الاتصال المنشأ
- تخزين البيانات ضمن غرض RecordSet لاستعادتها
- يمكن لهذا السيناريو أن يتم باستخدام الغرض Command أو عن طريق غرض Connection مباشرة
- لإدراج أو تعديل البيانات يمكن ببساطة تنفيذ عبارة SQL أو إجرائية مخزنة باستخدام الغرض Connection والغرض Command دون استخدام الغرض Recordset

يعتمد الوصول إلى البيانات في NET. على نفس الخطوط العريضة ولكن باستخدام مجموعة أخرى من الأغراض. قد تبدو هذه الأغراض مشابهة لكنها مختلفة بشكل جذري داخلياً مع اختلاف في الأداء ومرونة أكبر. إذ تعتمد أغراض الوصول إلى البيانات في NET. على غرضين أساسيين الأول DataReader والثاني هو DataSet . ينفذ كلا الغرضان العمل الذي كان الغرض Recordset يقوم به.

يكمن الفرق الرئيسي في أن الغرض DataReader يساعد في الوصول إلى البيانات باتجاه واحد وللقراءة فقط. في حين يوفر الغرض Dataset آلية للتعامل مع أكثر من مجموعة من الصفوف من نفس مصدر البيانات، حيث يمكننا إنشاء غرض DataSet من بيانات موجودة ضمن مصدر البيانات، أو من ملئها بصورة مباشرة صف تلو الآخر باستخدام النص البرمجي.

يحتوي كل جدول ضمن الغرض DataSet على معلومات حول القيم الأصلية للبيانات أثناء عملنا عليها، بحيث يمكن إرسال أي تعديلات على البيانات إلى مخزن البيانات في وقت لاحق.

يحتوي الغرض DataSet يحتوي معلومات تصف محتوي الجداول، كأنماط الأعمدة، القواعد، والمفاتيح .
يجب أن نتذكر دائماً أن التركيز في الغرض DataSet هو القدرة على العمل بصورة دقيقة وفعالة في بيئة غير متصلة .

يحافظ الغرض DataSet على محتوياته ويستطيع تحميل معلومات من وثيقة XML الحاوية على بيانات مهيكلة بالتنسيق الصحيح.

أغراض ADO.NET الأساسية

أغراض Connection:

يشبه هذا الغرض بصورة كبيرة الغرض المستخدم مع ADO ويحوي خصائص مشابهة له. يستخدم هذا الغرض لوصل غرض Command بمخزن البيانات .

- يُستخدم الغرض OleDbConnection مع مزود OLE_DB
- يستخدم الغرض SqlConnection ما يسمى TDS مع نظام إدارة قواعد البيانات SQLServer

كان من الممكن سابقاً لغرض Connection تنفيذ تعليمات SQL على مصدر البيانات أو فتح غرض

RecordSet. أما في ASP.NET فهذا غير ممكن. على كل الأحوال، تؤمن أغراض الاتصال السابقة الوصول إلى المناقلات التي تكون قيد التنفيذ على مخزن بيانات معين.

الطرق الأساسية لأغراض الاتصال:

من أهم الأغراض المستعملة لكل من غرضي الاتصال OleDbConnect و SqlConnection

الطريقة	الوصف
Open	تقوم بفتح اتصال إلى مصدر البيانات باستخدام الإعدادات الحالية مثل ConnectionString التي تحدد معلومات الاتصال المطلوب استخدامه.
Close	تقوم بإغلاق الاتصال الحالي مع مصدر البيانات.
BeginTransaction	تقوم ببدأ مناقلة مع مصدر البيانات وتعيد غرض Transaction الذي يمكن استخدامه لإتمام أو إلغاء المناقلة.

مثال:

```
' SQL Server Provider Sample
Dim connStr As String = "data source=(local);" & _
"initial catalog=Northwind;user id=sa"
Dim cnn As New SqlConnection(connStr)
cnn.Open()
' Use the connection in here…
If cnn.State = ConnectionState.Open Then
cnn.Close()
End If
```

أغراض ADO.NET الأساسية

أغراض الأوامر:

تشبه هذه الأغراض مثيلاتها Command في ADO وتمتلك نفس الخصائص، وتُستخدم لوصل غرض Connection مع غرض DataReader وغرض DataAdapter

- يستخدم الغرض OleDCommand مع مزود OLE-DB
- يستخدم الغرض SqlCommand مع خدمات البيانات الجدولية في SQLServer
- يسمح الغرض Command بتنفيذ عبارات SQL أو إجرائيات مخزنة على مصدر البيانات. يتضمن هذا إعادة مجموعة صفوف (حيث نستخدم للوصول إليها غرض آخر كالغرض DataReader أو كالغرض DataAdapter) أو إعادة قيمة وحيدة، أو إعادة عدد العناصر المتأثرة بالاستعلامات التي لا تعيد مجموعة صفوف

الطرق الرئيسية الخاصة بأغراض الأوامر:

الطريقة	الوصف
ExecuteNonQuery	تقوم بتنفيذ الأوامر المعرفة في الخاصة CommandText والمرتبطة باستخدام الاتصال المعرف بالخاصة Connection باستعلام لا يعيد أي صفوف (كتعليمة Update و Delete أو Insert). تجري إعادة رقم صحيح يعبر عن عدد الصفوف التي تأثرت بهذا الاستعلام.
ExecuteReader	تقوم بتنفيذ الأمر المعرف في الخاصة CommandText باستخدام الاتصال المعرف في الخاصة Connection. تعيد هذه الطريقة غرض reader متصل مع مجموعة الصفوف في قاعدة المعطيات بشكل يسمح بالحصول على الصفوف.
ExecuteScalar	تقوم بتنفيذ الأمر المعرف في الخاصة CommandText باستخدام الاتصال المحدد في الخاصة Connection. يعيد قيمة وحيدة هي العمود الأول من السطر الأول من مجموعة الصفوف التي يعيدها الاستعلام، ويتم إهمال جميع القيم الأخرى المعادة. هذه الطريقة سريعة وفعالة عندما يكون المطلوب إعادة قيمة وحيدة فقط.

مثال:

```
' OLE DB Provider Sample
Dim strSQL As String = "SELECT * FROM customer"
Dim cmd As New OleDbCommand(strSQL ,cnn)
```

```
' SQL Server Provider Sample
Dim strSQL As String = "SELECT * FROM customers"
Dim cmd As New SqlCommand(strSQL ,cnn)
```

أغراض ADO.NET الأساسية

أغراض DataAdapter:

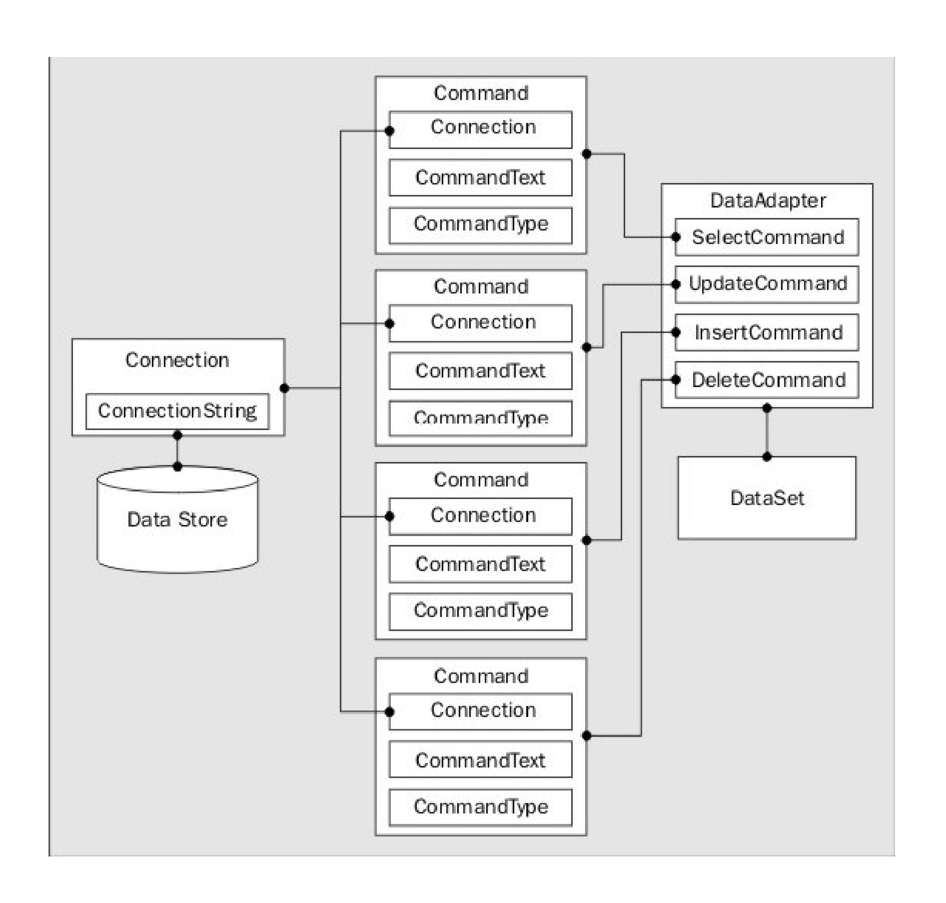
يعتبر هذا النوع من الأغراض نوعاً جديداً يقوم بوصل غرض Command أو أكثر بغرض DataSet

- الغرض OleDbDataAdapter المستخدم مع مزود OLE-DB
- الغرض SqlDataAdapter المستخدم مع خدمات البيانات الجدولية الخاصة ب MS SQl Server

توفر هذه الأغراض أربعة خصائص تُعرِّف الأوامر المستخدمة للتعامل مع البيانات في مخزن البيانات:

SelectCommand و InsertCommand و UpdateCommand و DeleteCommand

حيث تشكل كل من هذه الخصائص مؤشر إلى غرض Command (يمكن لكل هذه الأغراض الاشتراك بغرض Connection وحيد).



أغراض ADO.NET الأساسية

الطرق الأساسية التي تدعمها أغراض DataAdapter :

تقدم كل من أغراض OleDbDataAdapter و SqlDataAdapter مجموعة من الطرق للعمل مع أغراض DataSet التي تطبق عليها.أهم تلك الطرق ثلاثة هي:

الطريقة	الوصف
Fill	تقوم بتنفيذ الأمرSelectCommand لتأهيل الغرض DataSet بالبيانات من مصدر البيانات. كما يمكن أيضاً استخدامها لتحديث

	بالتعديلات DataSetالمعلومات المتوفرة ضمن جدول في غرض التي تمت على البيانات في مصدر البيانات الأصلي إذا كان هناك مفتاح رئيسي يميز صفوف البيانات في الجدول ضمن غرض Dataset.
FillSchema	لاستخلاص البنية الهيكلية لجدول SelectCommandتستخدم الأمر مع DataSetمن مصدر بيانات وتقوم بإنشاء جدول فارغ في غرض جميع القيود التي تحددها تلك البنية.
Update	تقوم باستدعاء الأوامر InsertCommand ،UpdateCommand ، DeleteCommand لكل صف تم تعديله أو حذفه من الغرض ، بحيث يجري تحديث هذه البيانات ضمن مصدر البيانات DataSet الأساسي، باستخدام التعديلات التي تمت على محتوى الغرض DataSet. المستخدمة مع غرض UpdateBatchتشبه هذه الطريقة، الطريقة لا يتم التعديل على ADOولكن في حالة ADO في Recordset أكثر من جدول.

مثال:

```
Dim connStr As String = "Provider=VFPOLEDB.1;Data Source=" & _
"C:\SAMPLES\DATA\TESTDATA.DBC"

Dim strSQL As String = "SELECT * FROM Products"

Dim oda As New OleDbDataAdapter(strSQL, connStr)

Dim cmdInsert As New OleDbCommand("INSERT INTO Products" & _
"(product_id, prod_name) VALUES (10,'car')")

oda.InsertCommand = cmdInsert
```

أغراض ADO.NET الأساسية

الغرض DataSet:

يقدم الغرض DataSet أساسيات التعامل مع قواعد البيانات العلائقية ووسائل تخزين البيانات في بيئة غير متصلة.

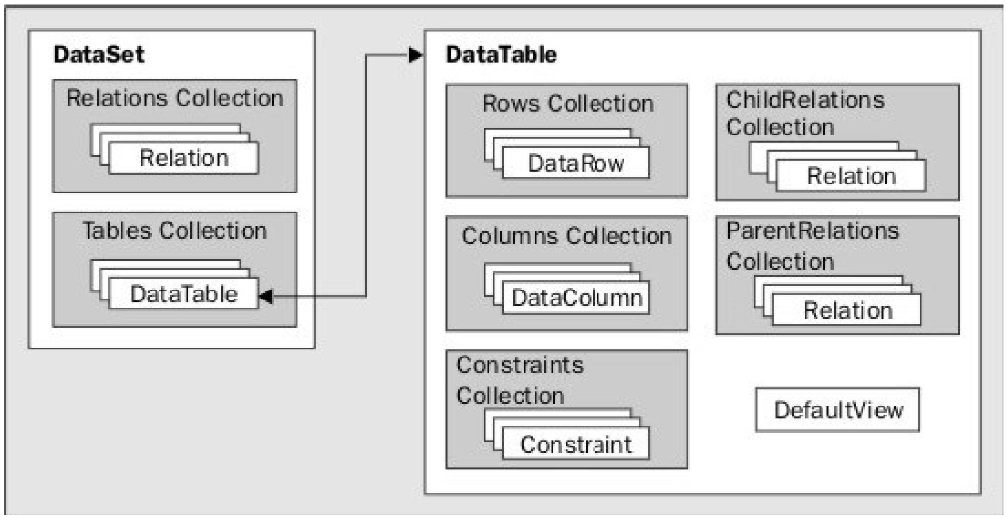
نقوم بتأهيل هذا الغرض من مخزن بيانات، ثم نقوم بالتعامل معه في حالة غير متصلة مع هذا المخزن، ثم نقوم بالاتصال وإتمام التغييرات على مخزن البيانات حسب الحاجة.

تتلخص الفروق الأساسية بين الغرض DataSet والغرض RecordSet (في ADO) بمايلي:

- يمكن لغرض DataSet أن يتعامل مع أكثر من مجموعة صفوف ويحتوي معلومات تخص العلاقة بينها، في حين لا يقدم الغرض RecordSet سوى إمكانية التعامل مع مجموعة صفوف واحدة.
- يوفر الغرض DataSet وصول غير متصل إلى البيانات بصورة آلية.

يكون كل جدول في الغرض DataSet عبارة عن غرض من النوع DataTable ينتمي إلى المجموعة Tables. ويحتوي كل غرض DataTable على مجموعة من أغراض DataRow ومجموعة من أغراض DataColumn.

هناك أيضاً مجموعات خاصة بالقيود كقيد المفتاح الأساسي والقيم التلقائية والعلاقات بين الجداول .



و أخيراً لدينا الغرض DefaultView الخاص بكل جدول والمخصص لإنشاء غرض DataView على ذلك الجدول، وذلك لتقديم إمكانيات البحث في البيانات والتعامل معها أو ربطها بعنصر تحكم.

أغراض ADO.NET الأساسية

الطرق الأساسية الخاصة بغرض DataSet:

يقدم الغرض DataSet مجموعة من الطرق التي يمكن استخدامها للتعامل مع محتويات الجداول أو للتعامل مع العلاقات القائمة فيما بينها، مثل عمليات مسح محتوى غرض DataSet أو عملية دمج محتويات أكثر من غرض DataSet:

الطريقة	الوصف
Clear	تقوم بإزالة جميع البيانات المخزنة في الغرض DataSet وذلك بتفريغ جميع الجدوال التي تحتويها. تكون عملية تدمير الغرض وإعادة إنشاؤه أكثر فعالية، في بعض الأحيان) إلا في حال الرغبة بالمحافظة على مؤشر لهذا العنصر.
Merge	تقوم هذه الطريقة بدمج محتوى غرض DataSet مع محتوى غرض DataSet آخر منتجةً غرض DataSet يحتوي جميع البيانات من كلا الغرضين.

طرق DataSet الخاصة بالتعامل مع XML:

ذكرنا سابقاً التنسيق التلقائي الذي تعتمده NET. هو تنسيق Xml عند تخزين البيانات. لذلك يقدم الغرض DataSet مجموعة من الطرق لقراءة وكتابة بيانات من وإلى وثائق XML.

الطريقة	الوصف
ReadXml ReadXmlSchema	تقوم بقراءة معلومات وثيقة XML أو هيكل XML وتأهيل غرض DataSet بها.
GetXml و GetXmlSchema	تعيد سلسة محرفية تحتوي وثيقة XML أو هيكل XML الذي يمثل البيانات في غرض DataSet.
WriteXml و WriteXmlSchem	تقوم بكتابة وثيقة XML أو هيكل XML الذي يمثل البيانات ضمن غرض DataSet إلى ملف على القرص أو غرض Reader/Writer

تقوم أغراض DataSet مع أغراض DataTable التي تحتويها بالاحتفاظ بسجل عن قيم محتوياتها عند إنشاءها أو تحميلها إذ يعتبر هذا الأمر مطلب أساسي مهم للسماح بتثبيت التغيرات في مخزن البيانات الرئيسي وبالأخص في بيئة متعددة المستخدمين.

مثال:

```
Dim connStr As String = "Provider=VFPOLEDB.1;Data Source=“ & _
“C:\SAMPLES\DATA\TESTDATA.DBC"

Dim strSQL As String = "SELECT * FROM Products"
Dim oda As New OleDbDataAdapter(strSQL, connStr)
Dim ds As New DataSet()
Dim dr As DataRow

oda.Fill(ds, "ProductInfo")
```

```
For Each dr In ds.Tables("ProductInfo").Rows
lstDemo.Items.Add(dr("Prod_Name").ToString)
Next dr

' Want to bind a Windows Form grid? It's this easy:
dgdDemo.DataSource = ds.Tables("ProductInfo")
```

أغراض ADO.NET الأساسية

يقدم الغرض DataSet أربعة طرائق لإعطاء قدرة أكبر على التحكم في كيفية ولحظة تثبيت البيانات:

AcceptChanges	تقوم بإتمام وتثبيت جميع التغييرات الحاصلة على الجداول والعلاقات ضمن غرض DataSet منذ أن تم تحميلها أو منذ آخر مرة تم تنفيذ الطريقة AcceptChanges فيها.
GetChanges	تعيد هذه الطريقة غرض DataSet يحتوي بعض أو كل التغيرات التي تمت منذ تم تحميل الغرض أو منذ آخر مرة تم فيها تنفيذ الطريقة AcceptChanges.
HasChanges	تحدد فيما إذا تمت أي تعديلات على البيانات منذ تحميل الغرض أو منذ آخر مرة تم تنفيذ الطريقة AcceptChanges فيها.
RejectChanges	تقوم بإهمال جميع التعديلات التي تمت على القيم في الجداول ضمن غرض DataSet منذ تحميل الغرض أو منذ آخر مرة تم تنفيذ الطريقة AcceptChanges فيها. تعيد هذه الطريقة المعلومات إلى الوضع الأصلي وتزيل جميع المعلومات المخزنة عن التغييرات.

الغرض DataTable:

يمكن الوصول إلى الجداول المحتواة في غرض DataSet عن طريق الغرض DataTable، كل غرض DataTable يمتلك خاصة تسمى Rows تؤشر إلى غرض DataRowCollection و هو عبارة عن مجموعة أغراض DataRow.

الطرق الأساسية لأغراض DataTable:

يوفر الغرض DataTable مجموعة من الخصائص والطرق التي تسمح بالتعامل مع كل جدول من جداول غرض DataSet على حدى. من أكثر الطرق استخداماً هي الطرق Clear و AcceptChanges و RegectChanges وهي مطابقة لتلك التي يدعمها الغرض DataSet إلا أنها تطبق فقط على جدول وحيد هو الجدول الذي يؤشر إليه الغرض DataTable. كما أن هناك مجموعة من الطرق الأخرى التي تسمح بالتعامل مع محتويات الجدول وهي:

الطريقة	الوصف
NewRow	تقوم بإنشاء صف في الجدول.حيث يتم إدخال القيم إلى الصف باستخدام النص البرمجي .
Select	تقوم بإعادة مجموعة من الأسطر التي تطابق تعبير تصفية معين.

مثال:

```
<%@ Page Language="vb" %>
<%@ import Namespace="System.Data" %>
<%@ import Namespace="System.Data.OleDb" %>
<script runat="server">

Sub Page_Load(sender As Object, e As EventArgs)
'declarations
Dim mycon As OleDbConnection
Dim mycmd As OleDbCommand
Dim mydap As OleDbDataAdapter
Dim mydst As DataSet

mycon = New OleDbConnection( _
"Provider=Microsoft.Jet.OLEDB.4.0;" & _
"Data Source=c:\myDb.mdb")

mydap = New OleDbDataAdapter ( _
"select col1,col2 from myTable", _
mycon)
mydst = New DataSet()
mydap.Fill(mydst,"col2")
mygrid.DataSource = mydst.Tables("col2")
mygrid.DataBind

End Sub

</script>
<html>
<head>
<title>Using DataAdapter - in  dot net</title>
```

```
</head>

<body bgcolor="#FFFFFF" text="#000000">
<asp:DataGrid id="mygrid"
Font-Name="Arial" Font-Size="10pt" BackColor="#CCCCCC"
Width="90%" Border="0" Cellpadding="5"
runat="server" AutoGenerateColumns="False">
<HeaderStyle font-bold="True" HorizontalAlign="Center"/>
<Columns>
<asp:BoundColumn DataField="col1" HeaderText="col1" />
<asp:BoundColumn DataField="col2" HeaderText="col2" />
/>
</Columns>
</asp:DataGrid>

</body>
</html>
```

أغراض ADO.NET الأساسية

الطرق الأساسية لغرض DataRowCollection:

يمثل هذا الغرض كما ذكرنا مجموعة من الصفوف ضمن غرض DataTable تم تحديدها بالخاصة Rows لهذا الغرض.

يوفر غرض DataRowCollection مجموعة من الطرق لإضافة أو إزالة صفوف ولإيجاد صف بناء (Constructor) على قيمة المفتاح الأساسي للجدول.

الطريقة	الوصف
Add	تقوم هذه الطريقة بإضافة سجل جديد تم إنشاؤه بالطريقة NewRow (الخاص بDataTable) إلى الجدول .
Remove	تقوم بإزالة أغراض DataRow المحددة من الجدول
Remove At	تقوم بإزالة سطر محدد بقيمة دليل موقعه في الجدول
Find	تقوم هذه الطريق بأخذ مصفوفة من قيم المفتاح الرئيسي وتعيد الصفوف المقابلة لها في الجدول كأغراض DataRow.

الطرق الأساسية لغرض DataRow:

هذا الغرض يمثل الصف بحد ذاته ضمن الجدول وضمن الغرض DataRowCollection. يمتلك هذا الغرض طرق AcceptChanges و RejectChanges والتي تقوم بنفس العمل الذي تقوم به مثيلاتها في

الغرض DataTable. إضافة إلى هذه الطرق يقدم هذا الغرض مجموعة من الطرق المستخدمة للتعامل مع البيانات في صف واحد من الجدول.

الطريقة	الوصف
BeginEdit, EndEdit, CancelEdit	تستخدم لتحويل الصف إلى وضعية التحرير وحفظ أو إلغاء التغييرات على البيانات في وضع التحرير.
Delete	تقوم بتمييز الصف كصف محذوف، ولكن لن تتم إزالته من الجدول لحين تنفيذ الطريقة AcceptChanges أو Update.
GetChildsRows	تقوم هذه الطريقة بإعادة مجموعة من الصفوف من جدول مرتبط بهذا الصف بعلاقة ابن.
SetColumnError وGetColumnsInError	تستخدم لتعيين وإعادة حالة الخطأ لهذا الصف. حيث تستخدم هذه الطرق مع الخصائص HasErrors و RowError لتحري أو لتحديد أماكن الخطأ.

مثال:

يقوم هذا المثال تقوم باستخدام الطريقة Find لإيجاد صف يحتوي قيمة محددة ثم يقوم بحذفه.

```
Private Sub RemoveFoundRow(ByVal table As DataTable)
Dim rowCollection As DataRowCollection = table.Rows

' Test to see if the collection contains the value.
If rowCollection.Contains(TextBox1.Text) Then
Dim foundRow As DataRow = rowCollection.Find(TextBox1.Text)
rowCollection.Remove(foundRow)
Console.WriteLine("Row Deleted")
Else
Console.WriteLine("No such row found.")
End If
End Sub
```

أغراض ADO.NET الأساسية

الغرض DataView:

كما ذكرنا مسبقاً يمكننا تأهيل غرض DataView من خلال جدول في غرض DataSet. يقوم الغرض DataView بالتعامل مع جدول أو مجموعة من الصفوف في جدول. يمكن إنشاءه باستخدام الغرض DefaultView الخاص بالجدول أو من خلال الغرض DataTable الذي يقوم باختيار مجموعة جزئية من الصفوف في الجدول.

الطرق الرئيسية للغرض DataView:

بصورة عامة إن أفضل طريقة للتعامل مع محتويات الجدول ضمن الغرض DataSet هي إنشاء غرض DataView من الجدول المراد استخدامه ثم استعمال الطرق التي يوفرها:

الطريقة	الوصف
AddNew	تقوم هذه الطريقة بإضافة صف جديد إلى الغرض DataView. يمكن بعدها استخدام النص البرمجي لإدراج قيم في هذا الصف.
Delete	تقوم هذه الطريقة بإزالة الصف محدد من الغرض DataView.
`Find	تأخذ هذه الطريقة كمعامل قيمة وحيدة أو مصفوفة من القيم وتعيد الدليل للصف الذي يطابق هذه القيم.
FindRows	تأخذ قيمة وحيدة أو مصفوفة من القيم وتعيد مجموعة من الأغراض DataRow التي تطابق هذه القيم.

مثال:

```
Dim strSQL As String = "SELECT * FROM Products"
Dim sda As New SqlDataAdapter(strSQL, connStr)
Dim ds As New DataSet()

sda.Fill(ds, "ProductInfo")

Dim dv As New DataView(ds.Tables("ProductInfo"))

dv.Sort = "Prod_Name ASC"

dv.RowFilter = "In_Stock > 100"

' Bind the DataGrid to this DataView.
Me.dgdDemo.DataSource = dv
```

أغراض ADO.NET الأساسية

الغرض DataReader:

توفر أغراض DataSet أرضية مناسبة للوصول إلى البيانات في بيئة غير متصلة، ولكن في الكثير من الحالات، وبالأخص في تلك التي نحتاج فيها إلى طريقة سريعة وفعالة للوصول إلى البيانات بدون الاضطرار إلى سحب جميع البيانات.

يعد استخدام الغرض DataReader من أنجع الحلول لتحقيق:

- سحب سجل أو أكثر أو سحب قيم من حقول محددة
- تنفيذ تعبيرات INSERT SQL، UPDATE،DELETE؛
- عندما يكون لدينا فيها كمية كبيرة من البيانات أكبر من أن تتسع لها أغراض DataSet
- ربط عناصر التحكم من جهة المخدم

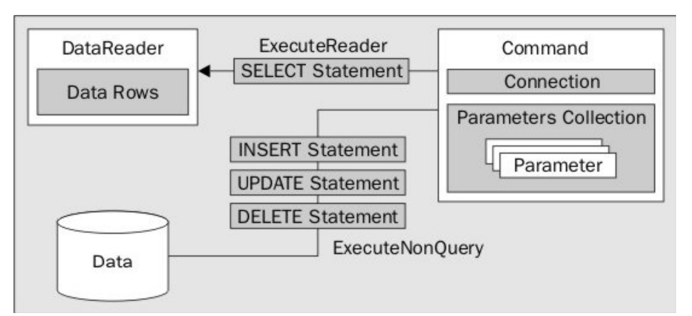
هناك نوعان أساسيان من الغرض DataReader:

- الغرض OleDbDataReader وهو يستخدم OLE-DB
- الغرض SqlDataReader الذي يستخدم خدمات البيانات الجدولية الخاصة بمخدم SQLServer

يقوم الغرض DataReader بتنفيذ تعليمة SQL أو تخزين إجرائيات مخزنة لاستحضار مجموعة من صفوف البيانات والتجوال فيما بينها إذ تتم المحافظة على الاتصال مع مخزن البيانات في ذلك الوقت

- يوفر الغرض DataReader شبيه للمؤشرات المستخدمة مع مخازن البيانات والتي تستخدم تعبيرات SQL أو الإجرائيات المخزنة لاستخلاص مجموعة الصفوف
- يوفر الغرض DataReader إمكانية تنفيذ تعليمات SQL أو إجرائيات مخزنة لتحديث البيانات
- لا يوفر هذا الغرض وصول غير متصل إلى البيانات

الوصول إلى مجموعة الصفوف التي يشير إليها الغرض DataReader هووصول مخصص للقراءة فقط وباتجاه واحد



أغراض ADO.NET

الطرق الأساسية للغرض DataReader:

لاستخدام غرض DataReader نقوم بإنشاء غرض Command ثم نستخدمه لتنفيذ تعبيرات SQL أو إجرائيات مخزنة وإعادة غرض DataReader.

يمكننا بعدها الدوران عبر هذه الصفوف والأعمدة باستخدام الغرض DataReader لاستخلاص النتائج من

مخزن البيانات.

أكثر طرق الغرض DataReader استخداماً هي التالية:

الطريقة	الوصف
Read	يقوم بدفع مؤشر الصف خطوة إلى الأمام ليؤشر إلى السطر التالي بحيث يتم الوصول إلى قيم الأعمدة باستخدام اسم العمود أو رقم التسلسل الخاص به.
GetValue	تعيد قيمة من الصف الحالي بنفس التنسيق المُستخدم في مصدر البيانات وذلك بتحديد دليل العمود. لتنفيذ عملية بشكل أبسط ولكن أكثر فعالية يمكن استخدام اسم العمود مباشرة وذلك بالشكل . Value=DataReader("col-Name")
GetValues	يقوم باعادة قيمة أو مجموعة قيم من الصف الحالي بتنسيقها الأصلي وذلك ضمن مصفوفة.
Getxxxxxx	تقوم بإعادة قيمة من السطر الحالي بتنسيق نمط البيانات المحدد حسب الطريقة xxxxxx. مثل GetBoolean,GetInt16,GetChars.
NextResult	تقوم هذه الطريقة بنقل مؤشر الصف إلى مجموعة الصفوف الأخرى عندما تقوم تعليمة SQL أو الإجرائية المخزنة بإعادة أكثر من مجموعة صفوف. تجدر الإشارة إلى أن هذه الطريقة ليست مشابهة للطريقة MoveNext لأنها لاتنتقل ضمن نفس مجموعة الصفوف بل من مجموعة صفوف إلى أخرى.
Close	تقوم بإغلاق الغرض DataReader وتحرير المؤشر الذي يشير إلى مجموعة الصفوف.

أغراض ADO.NET

متى يجب استخدام غرض DataReader ومتى يجب استخدام غرض DataSet:

عند البدء ببناء تطبيقاتنا للوصول إلى مخازن البيانات، يجب علينا أن نفكر بنوع الوصول الذي نريد وكيف سيتم استعمال البيانات.

لا بد وأنه أصبح من الواضح مما ذكرناه مسبقاً أن أغراض DataSet تسبب حمل وتعقيد لا يمكن إهماله مقارنة مع الغرض DataReader من حيث الأداء واستهلاك الذاكرة. إذاً لابد لنا من التركيز على استخدام الغرض DataReader عوضاً عن DataSet. إلا أن هناك بعض الحالات التي لا يمكن فيها الاستغناء عن DataSet وهي:

- عند حاجتنا إلى استخدام البيانات بشكل غير متصل مع مخزن البيانات وتمرير هذه البيانات إلى أطر أخرى ضمن التطبيق، وتخزينها ، وتحريرها
- عند الحاجة إلى تخزين، ونقل، والوصول إلى أكثر من جدول (أكثر من غرض DataTable) ومعالجة العلاقات بين هذه الجداول
- عند الحاجة إلى تحديث البيانات في قاعدة البيانات باستخدام طرق خاصة بغرض DataSet و DataAdapter عوضاً عن استخدام تعبيرات SQL UPDATE أو استخدام الإجرائيات المخزنة
- عند الحاجة لإدارة الوضع بشكل أفضل في بيئة متعددة المستخدمين
- عندما نريد الاستفادة من المزامنة بين وثائق XML ومجموعة الصفوف العلائقية المقابلة
- في بعض حالات الربط مع عناصر التحكم مثل حالات الربط مع أكثر من عنصر في نفس الوقت، أو تقسيم السجلات إلى صفحات في عنصر تحكم DataGrid، إذ نستخدم في هذه الحالة الغرض DataView المنشأ من جدول في غرض DataSet
- في حال أردنا الدوران ضمن صفوف البيانات وأردنا الحصول على حرية في اتجاه الحركة للأمام أو الخلف. هنا أيضاً لا يمكننا استخدام الغرض DataReader

مثال:

```
SqlConnection conn = new SqlConnection(connectionString);
 SqlCommand comm = new SqlCommand("select * from mytable", conn);
 comm.Connection.Open();
 SqlDataReader r =
    comm.ExecuteReader(CommandBehavior.CloseConnection);
 while(r.Read())
```

```
  Console.WriteLine(r.GetString(0));
 }
 r.Close();
 conn.Close()
```

مزودات البيانات العلائقية في NET.

كما رأينا سابقاً تستخدم NET. مزودات بيانات للاتصال بمخازن البيانات . يوفر إطار عمل NET. دعم لمجموعة المزودات التالية:

اسم المزود	الوصف
SQLOLEDB	مزودOLE-DB ل SQLServer
MSDAORA	مزودOLE-DB ل ORACLE
Microsoft.Jet.OLEDB.4.0	مزودOLE-DBل Access مصادر البياناتJet التي تستخدم مزود

تقدم NET. أيضاً المزود ODBC الذي تم تطويره ليسمح بالاتصال بالعديد من التجهيزات ومخازن البيانات، وهو يستخدم أغراض مشابهة لتلك المستخدمة مع المزودات الأخرى فهو يتضمن مثلاً OdbcConnection، OdbcCommand، OdbcDataReader، OdbcDataAdapter...إلخ.

أمثلة

مثال استخدام الغرض OleDbConnection

```
<%@ Import Namespace="System.Data.OleDb" %>
<script runat="server">
sub Page_Load
dim dbconn
dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
data source=" & server.mappath("northwind.mdb"))
dbconn.Open()
end sub
</script>
```

مثال استخدام الغرض Command

```
<%@ Import Namespace="System.Data.OleDb" %>
<script runat="server">
sub Page_Load
dim dbconn,sql,dbcomm
```

```
dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
data source=" & server.mappath("northwind.mdb"))
dbconn.Open()
sql="SELECT * FROM customers"
dbcomm=New OleDbCommand(sql,dbconn)
end sub
</script>
```

مثال استخدام الغرض DataReader

```
<%@ Import Namespace="System.Data.OleDb" %>
<script runat="server">
sub Page_Load
dim dbconn,sql,dbcomm,dbread
dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
data source=" & server.mappath("northwind.mdb"))
dbconn.Open()
sql="SELECT * FROM customers"
dbcomm=New OleDbCommand(sql,dbconn)
dbread=dbcomm.ExecuteReader()
end sub
</script>
```

مثال استخدام غرض DataSet و DataAdapter و DataView

```
<%@Page Language="VB"%>

<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.OleDb" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>The .NET DataSet and OleDbDataAdapter Objects</title>
<!-- #include file="..\global\style.inc" -->
</head>
<body bgcolor="#ffffff">
<span class="heading">The .NET DataSet and OleDbDataAdapter
Objects</span><hr />
<!-----------------------------------------------------------------
---------->

<div>Connection string: <b><span id="outConnect"
runat="server"></span></b></div>
<div>SELECT command: <b><span id="outSelect"
runat="server"></span></b></div>
<div id="outError" runat="server"> </div>

<asp:datagrid id="dgrResult" runat="server" />

<script language="vb" runat="server">
```

```
Sub Page_Load()

'get connection string from web.config file
Dim strConnect As String
strConnect = ConfigurationSettings.AppSettings("DsnWroxBooksJet") _
& Request.PhysicalApplicationPath _
& "databases\WroxBooks.mdb"
outConnect.innerText = strConnect 'and display it

'specify the SELECT statement to extract the data
Dim strSelect As String
strSelect = "SELECT * FROM BookList WHERE ISBN LIKE '07645437%'"
outSelect.innerText = strSelect   'and display it

'declare a variable to hold a DataSet object
'note that we have to create it outside the Try..Catch block
'as this is a separate block and so is a different scope
Dim objDataSet As New DataSet()

Try

'create a new Connection object using the connection string
Dim objConnect As New OleDbConnection(strConnect)

'create a new DataAdapter using the connection object and select
statement
Dim objDataAdapter As New OleDbDataAdapter(strSelect, objConnect)

'fill the dataset with data from the DataAdapter object
objDataAdapter.Fill(objDataSet, "Books")

Catch objError As Exception

'display error details
outError.innerHTML = "<b>* Error while accessing data</b>.<br />" _
& objError.Message & "<br />" & objError.Source
Exit Sub  ' and stop execution

End Try

'create a DataView object for the Books table in the DataSet
Dim objDataView As New DataView(objDataSet.Tables("Books"))

'assign the DataView object to the DataGrid control
dgrResult.DataSource = objDataView
dgrResult.DataBind()   'and bind (display) the data

End Sub
</script>
```

```
<!------------------------------------------------------------------
---------->
<!-- #include file="..\global\foot.inc" -->
</body>
</html>
```

مثال استخدام DataSet و DataTable و DataColumn و DataRow:

```
Dim ds As New DataSet()

Dim strSQL As String = "SELECT Cust_ID, Order_Id, “ & _
“Order_Date FROM Orders WHERE Year(Order_Date) > 1997"
Dim oda As New OleDbDataAdapter(strSQL, connStr)
oda.Fill(ds, "OrderInfo")

strSQL = "SELECT CustomerID FROM Customers “  & _
“ORDER BY CustomerID"
Dim sda As New SqlDataAdapter(strSQL, connStr)
sda.Fill(ds, "CustomerInfo")

Dim dc1 As DataColumn = _
ds.Tables("CustomerInfo").Columns("CustomerId")
Dim dc2 As DataColumn = _
ds.Tables("OrderInfo").Columns("Cust_Id")

Dim dr As New DataRelation("CustomersToOrders", dc1, dc2)
ds.Relations.Add(dr)

Dim drCustomer As DataRow
Dim drOrder As DataRow

For Each drCustomer In ds.Tables("CustomerInfo").Rows
lstDemo.Items.Add("Customer: " & _
drCustomer("CustomerId").ToString())
' Iterate through related rows.
For Each drOrder In drCustomer.GetChildRows(dr)
lstDemo.Items.Add( _
String.Format("   Order {0} placed on {1:d}", _
drOrder("Order_ID"), drOrder("Order_Date")))
Next drOrder
Next drCustomer
```

مقدمة عن XML في NET.

تكلمنا سابقاً عن المزايا الجديدة في إطار العمل NET. والمخصصة للوصول إلى البيانات العلائقية وقمنا بمقارنة بسيطة مع التقنيات التقليدية التي كانت مستخدمة مع ADO.

أصبحت لغة XML وبسرعة، اللغة الأساسية على الوب وتم اعتمادها من قبل العديد من التطبيقات. لذا سنركز في هذا الجزء على الطريقة التي تدعم بها NET. لغة XML

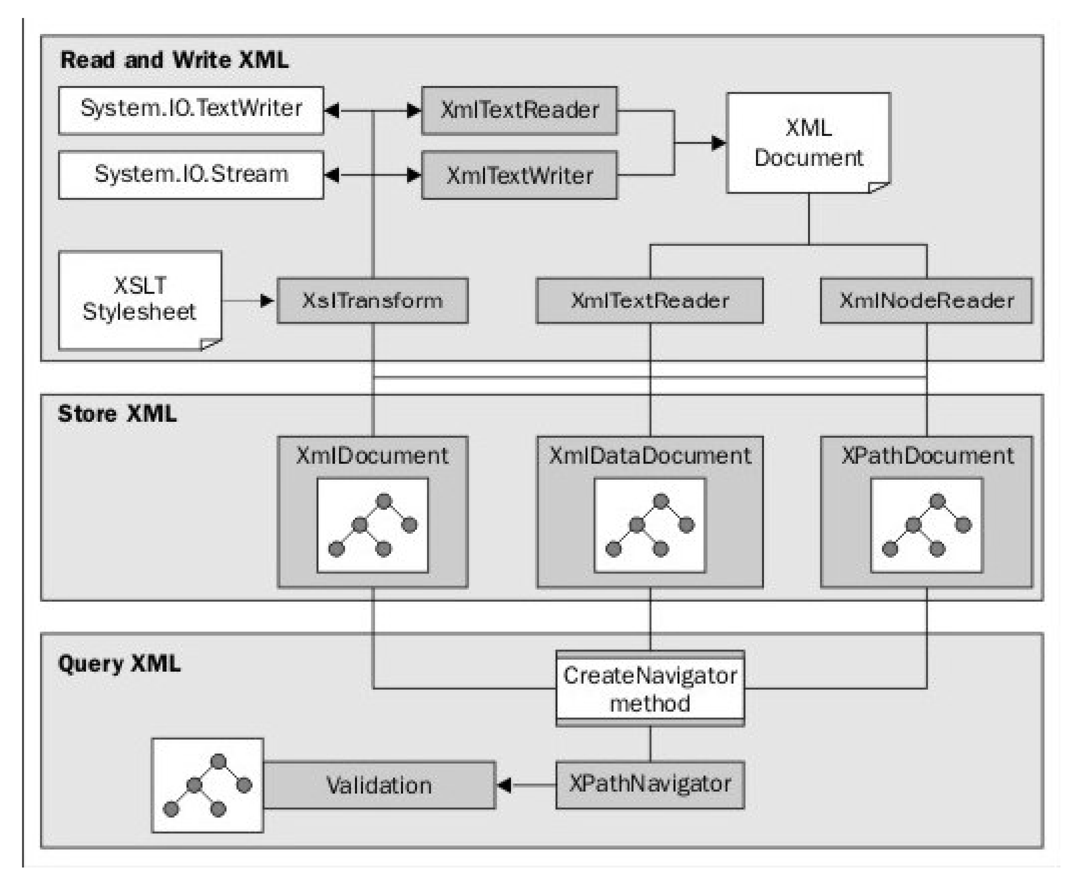
أغراض XML الأساسية في NET.:

قدمت منظمة W3C مجموعة من المعايير التي حددت البنية والواجهات التي يجب تزويد التطبيقات بها للوصول إلى وثائق XML. تسمى هذه المعايير بـــ (DOM). وقد تبنت NET. دعم هذه النعايير من خلال الأغراض XmlDocument و XmlDataDocument.

توفر هذه الأغراض دعماً كاملاً لمعايير XML (DOM) من الدرجة الثانية.

تقديم إلى XML في NET.

وسعت NET. دعمها لـXML عبر تقديم تقنيات للتعامل مع وثائق XML، وهياكل XML وملفات الأنماط . فيما يلي مخطط يوضح الأغراض التي نستخدمها للتعامل مع وثائق XML:



تتوزع أغراض XML الأساسية في ثلاث مجموعات:

- الأولى لقراءة وكتابة وتحويل ملفات XML، وتتضمن هذه المجموعة أغراض XmlTextReader و XmlTextWriter و XmlNodeReader إضافة إلى غرض XslTransform الخاص بإنشاء ملف بتنسيق مختلف عن وثيقة XML الأصلية.
- الثانية خاصة بتخزين وثائق XML والتعامل معها وهي تتضمن الأغراض مثل XmlDocument، XmlDataDocument، XPathDocument.
- أما الثالثة فهي خاصة باستعلام XML ونستخدم فيها الغرض XPathNavigator.

بالطبع سنلاحظ تداخل بين هذه الوظائف. فلاختبار وثيقة XML أثناء قراءتها، نستخدم الغرض XmlValidatingReader وهناك أغراض أخرى لإنشاء وتحرير هياكل ومخططات XML. كما يمكننا أيضاً استخدام الغرض XslTransform لتنفيذ استعلام على وثيقة بالإضافة إلى تحويلها إلى تنسيق مختلف.

أغراض Document

لدينا ثلاث آليات تطبيقية لغرض Document:

• الغرض XmlDocument: وهو عبارة عن آلية تطبيق Net. لمعايير Dom. تتضمن خصائص وطرق هذا الغرض تلك المعرفة من قبل W3C للتعامل مع وثائق XML مع بعض الإضافات لجعل العمليات الأساسية أسهل.

• الغرض XmlDataDocument: هو عبارة عن امتداد للغرض XmlDocument يوفر نفس مجموعة الخصائص والطرق، كما يعمل كجسر وصل بين XML وطرائق قواعد البيانات العلائقية.

• الغرض XPathDocument يمثل آلية سريعة لتخزين وثائق XML، وهو مصمم خصيصاً ليتم الوصول إليه باستخدام الغرض XPathNavigator، باستعمال استعلامات XPath فقط أو الملاحة ضمن الوثيقة عنصر تلو الآخر باستخدام تقنية "Pull".

الطرق الأساسية لأغراض Document:

لا يمتلك الغرض XPathDocument طرق عامة ذات فائدة غير الطريقة CreateNavigator كونه مصمم ليعمل فقط مع الغرض XPathNavigator. في حين يوفر النوعان الآخران لغرض Document مجموعة كاملة من الطرق والخصائص وهي تلك المحددة في معايير W3C DOM. إضافة لما ذكر يوفر هذان النوعان مجموعة مفيدة من الطرق والخصائص التي تستخدم بشكل كبير للعمل مع وثائق XML وهي تتضمن طرق لإنشاء أنواع معينة من العقد والوصول إلى عقد موجودة.

الطريقة	الوصف
Createxxxxx	تقوم بإنشاء عقدة في وثيقة XML اعتماداً على اسم الطريقة، مثل CreateElement، CreateComment، CreateTextNode...إلخ
CloneNode	تقوم بإنشاء نسخة مثيلة عن عقدة XML مثل نسخة من عقدة Element
GetElementById	تقوم بإعادة عقدة وحيدة باستخدام قيمة الواصفة ID كمعامل.
GetElementsByTagname	تقوم بإعادة عقدة وحيدة باستخدام اسم التأشيرة كمعامل

هناك أيضاً مجموعة من الطرق لتحميل وحفظ XML من وإلى غرض وثيقة XML.

Save	تقوم بحفظ وثيقة XML كاملة إلى ملف على القرص أو إلى غرض Stream أو إلىغرض XmlTextWriter.
Read Node	تقوم بتحميل عقدة من وثيقة XML يجري التأشير إليها بغرض XmlTextReader أو غرض XmlNodeReader.
Write To	تقوم بكتابة عقدة إلى وثيقة XML يجري التأشير إليها بغرض XmlTextWriter
Write ContentTo	تقوم بكتابة عقدة وكل ما يندرج تحتها إلى وثيقة XML أخرى يجري التأشير إليها بغرض XmlTextWrite.

الطرق الأساسية لأغراض Document (متابعة)

أما إذا أردنا استعمال الغرض XPathNavigator في وثيقتنا فإننا نقوم بإنشاءه باستخدام الطريقة CreateNavigator:

الطريقة	الوصف
CreateNavigator	تقوم بإنشاء وإعادة غرض XPathNavigator مبني على وثيقة XMLالحالية. يمكن تطبيقه على جميع أنواع أغراض Document . ومع XmlDocument و XmlDataDocumentتقبل هذه الطريقة معامل إضافي يشكل مؤشر إلى عقدة في الوثيقة والتي ستكون موقع البداية لغرض XPathNavigator

يدعم غرض XmlDataDocument خاصة إضافية عن الخصائص يقدمها الغرض XmlDocument :

الخاصة	الوصف
DataSet	تقوم بإعادة محتوى وثيقة XML كغرض DataSet.

كذلك يدعم الغرض XmlDataDocument طريقتين إضافيتين توفران وصول أقوى إلى محتوى الوثيقة وذلك بالتعامل معها كمجموعة صفوف أو جدول بيانات.

الطريقة	الوصف
GetRowFromElement	يعيد غرض DataRow يمثل العناصر في الوثيقة.
GetElementFromRow	يعيد غرض XmlElement يمثل DataRow في جدول ضمن DataSet

الغرض XPathNavigator

لجعل العمل مع وثائق XML أسهل تم تعريف الغرض XPathNavigator في فضاء الأسماء System.Xml والذي يمكن استخدامه للتجوال ضمن وثيقة XML أو للاستعلام عن مكون ضمن الوثيقة باستخدام تعبير XPath .

نشير إلى أنه يمكننا استخدامه الغرض XPathNavigator مع أي غرض وثيقة XML وليس فقط مع XPathDocument. أي يمكننا إنشاء غرض XPathNavigator مبني على غرض XMLDocument أو على غرض XmlDataDocument.

يقدم الغرض XPathNavigator مجموعة من الطرق والخصائص التي تسمح بالتجول ضمن وثيقة XML كالانتقال بين العقد بالترتيب أو بتجاوز العقد حتى الوصول إلى عقدة من نمط معين.

يقدم الغرض XPathNavigator مجموعة من الطرق التي تقبل تعبيرات XPath، أواسم العقدة، أونمط العقدة، وتعيد العقدة أو مجموعة العقد المطابقة. عندها نستطيع التجوال عبر هذه العقد.

يمكن إنشاء الغرض XPathNavigator فقط من غرض وثيقة موجود وذلك كمايلي:

```
Dim objNav1 As XPathNavigator = objXMLDoc.CreateNavigator()
Dim objNav2 As XPathNavigator = objXMLDataDoc.CreateNavigator()
Dim objNav3 As XPathNavigator = objXPathDoc.CreateNavigator()
```

الطرق الأساسية لغرض XPathNavigator

يسمح هذا الغرض بالتجوال عبر الوثيقة واختيار عقدة ضمن وثيقة XML والوصول إليها. يمكننا إنشاء أكثر من غرض من هذا النوع على نفس الوثيقة ومقارنة مواقعها.

لتحرير وثيقة XML نستخدم مؤشر إلى العقدة الحالية لهذا الغرض أو غرض XPathNodeIterator الذي يحتوي أكثر من مجموعة من العقد، ونقوم باستدعاء الطرق الخاصة بعقدة ما من المجموعة.

في نفس الوقت يوفر الغرض XPathNavigator تفاصيل حول العقدة الحالية وبهذا يكون لدينا طريقتان للوصول إلى معلومات عقدة ما ضمن الوثيقة.

فيما يلي جدول يوضح أكثر طرق الغرض XPathNavigator استخداماً. هناك طرق للحركة ضمن الوثيقة لجعل عقدة ما هي العقدة الحالية بالنسبة لغرض XPathNavigator أو لإنشاء غرض XPathNavigator جديد:

الطريقة	الوصف
MoveToxxxxxx	يقوم بتحريك موقع الغرض ضمن وثيقة XML الحالية. لدينا مثلاً: MoveToFirst ,MoveToRoot ,MoveToAttribute, moveToParent,MoveToFirstChild..
Clone	إنشاء غرض XPathNavigator جديد متوضع في نفس المكان الذي يتوضع فيه غرض الحالي XPathNavigator ضمن الوثيقة.

هناك أيضاً طرق مخصصة للوصول واختيار أجزاء من محتوى وثيقة:

الطريقة	الوصف
GetAttribute	تقوم بإعادة القيمة المحددة بالمعامل المحدد من العقدة الحالية للغرض.
Select	تعيد غرض XPathNodeIterator الذي يحتوي مجموعة من العقد التي تطابق التعبير XPath.
SelectAncestors	تعيد هذه الطريقة غرض XPathNodeIterator الحاوي على مجموعة من جميع عقد السلف في الوثيقة والتي لها نمط معين واسم معين.
SelectDescendants	تعيد غرض XPathNodeIterator يحتوي مجموعة من العقد المتحدرة من الوثيقة والتي لها نمط معين أواسم معين.
SelectChildre	تعيد غرض XPathNodeItarator يحتوي مجموعة من العقد

n	الأبناء من نمط معين أو اسم معين

الغرض XmlTextWriter

عند استخدام الغرض XmlDocument لإنشاء وثيقة XML جديدة يجب علينا إنشاء أجزاء وإدراجها ضمن الوثيقة بطريقة معينة، وهي طريقة قد تكون معقدة ومصدر للخطأ.
يمكن للغرض XmlWriter أن يُستخدم لإنشاء وثيقة XML عقدة تلو الأخرى بطريقة تسلسلية بكتابة التأشيرات والمحتوى إلى الخرج باستخدام مجموعة الطرق التي يقدمها هذا الغرض.
يأخذ الغرض XmlTextWriter كمصدر، إما غرض TextWriter الذي يوشر إلى ملف على القرص ،و مسار، واسم هذا الملف، أو غرض Stream الذي يحتوي وثيقة XML جديدة .

- يقدم هذا الغرض مجموعة من الخصائص والطرق التي يمكن استخدامها لإنشاء عقد XML ومحتويات أخرى، ثم يقوم بتوجيه الخرج إلى ملف على القرص أو غرض Stream بصورة مباشرة.
- يمكن أن يتم تعيين الغرض XmlTextWriter كأداة خرج لطرق في أغراض متعددة حيث يقوم غرض XmlTextWriter بتوجيه المحتوى إلى ملف على القرص أو غرض TextWriter أو غرض Stream

الطرق الأساسية لغرض XmlTextWriter: اكثر الطرق استخداماً لهذا الغرض هي التالية:

الطريقة	الوصف
WriteStartDocument	تقوم ببدأ وثيقة جديدة وكتابة تصريح XML إلى الخرج.
WriteEndDocument	تقوم بإنهاء الوثيقة بإغلاق جميع العناصر غير المغلقة ثم إرسال المحتوى إلى الخرج.
WriteStartElement	تقوم بفتح تأشيرة لعنصر محدد. عندها يمكن البدء بإنشاء الواصفات باستخدام الطريقة WriteStartAttribute
WriteEndElement	تقوم بكتابة تأشيرة إغلاق للعنصر الحالي. الطريقة المقابلة لإغلاق واصفة هي WriteEndAttribute.
WriteElementString	تقوم بكتابة عنصر كامل (بما يتضمن تأشيرات الفتح والإغلاق) باستخدام سلسة محارف كقيمة. الطريقة المقابلة لأداء نفس العلم مع الواصفات هي WriteAttributeString.
Close	تقوم بإغلاق الملف على القرص أو غرض Streamوتحرر جميع المؤشرات المرتبطة.

الغرض XmlReader

نحتاج في بعض الأحيان إلى قراءة وثائق XML من مصدر آخر عوضاً عن كتابتها.

يعتبر الغرض XmlReader صف قاعدي يرث منه صفان عامان هما XmlTextReader و XmlNodeReader.

- يعتمد الغرض XmlTextReader على الغرض TextReader الذي يؤشر إلى ملف XML على القرص، وإلى مساره واسمه أوإلى غرض Stream يحتوي وثيقة XML. يمكن قراءة محتوى الوثيقة عقدة تلو الأخرى، ويوفر الغرض معلومات حول كل عقدة وحول القيمة التي تحتويها أثناء قراءتها.
- يأخذ الغرض XmlNodeReader غرض XmlNode كمصدر له مما يسمح بقراءة جزء من وثيقة XML عوضاً عن قراءة كامل الوثيقة في حال كان هدفنا لوصول إلى عقدة معينة وإلى العقد الأبناء لهذه العقدة.
- يمكن أن يُستخدم الغرضان XmlTextReader و XmlNodeReader بصورة منفصلة لتوفير وصول سريع وفعال لوثائق XML أو كمصدر لأغراض أخرى حيث نقوم تلقائياً بقراءة الوثيقة و تمرير ناتج القراءة إلى الغرض الأب.

تستخدم الأغراض XPathNavigator و XmlTextReader نمذجة "pull" للوصول إلى البيانات عقدة تلو الأخرى عوضاً عن تفريغ كامل الوثيقة كشجرة ضمن الذاكرة. يسمح ذلك بالوصول إلى ملفات كبيرة الحجم دون استهلاك الكثير من الموارد ويجعل عملية كتابة النص البرمجي أسهل لإي أغلب الحالات.

كما تبرز أهمية هذه الأغراض في الحالات التي نبحث فيها عن قيمة معينة عندها لن نضطر إلى قراءة كامل الوثيقة بل يمكننا الوصول إلى عقدة محددة بعد قراءة جزء من الوثيقة.

الطرق الأساسية لغرض XmlReader

يمتلك الغرضان XmlTextReader و XmlNodeReader خصائص وطرق متطابقة تقريباً، أهم الطرق المستخدمة هي:

الطريقة	الوصف
Read	تقوم بقراءة العقدة التالية إلى الغرض حيث يمكن الوصول إليها. تعيد القيمة False عندما لا يتبقى أي عقد لقراءتها.
ReadInnerXml	تقوم بقراءة وإعادة محتوى العقدة بشكل سلسلة محارف بما فيها جميع التأشيرات والنصوص للعقد الأبناء.
ReadOuterXml	تقوم بقراءة وإعادة محتوى و**تأشيرات** العقدة الحالية كسلسلة محارف بما فيها جميع التأشيرات والنصوص للعقد الأبناء .
ReadString	تقوم بإعادة سلسلة محارف تحتوي قيمة العقدة الحالية.
GetAttribute	تقوم بإعادة قيمة الواصفة المحددة للعقدة الحالية لغرض XmlReader
GetRemainder	تقوم بقراءة وإعادة الجزء الباقي من وثيقة XML المصدرية كسلسلة محارف وتعد هذه الطريقة مفيدة عندما نرغب بنقل XML من وثيقة إلى أخرى.
MoveToxxxxxx	تقوم بنقل موقع مؤشر القارئ. مثال: MoveToAttribute و MoveToContent و MoveToElement
Skip	تقوم بتجاوز العقدة الحالية والتحرك إلى العقدة التالية.
Close	تقوم بإغلاق الملف على القرص أو إغلاق الغرض Stream الذي يتم التعامل معه.

الغرض XMLValidatingReader

هناك غرض آخر مبني على الصف XmlReader وهو الغرض XmlValidatingReader .

يمكننا النظر إلى هذا الغرض كغرض XmlTextReader ولكن بمهمة تقييم وثيقة مقارنةً بهيكل أو بمخطط ما.

يمكننا إنشاء غرض XmlValidatingReader من غرض XmlReadr موجود، أو من غرض Stream، أو من سلسلة محارف تحتوي XML ليتم تقييمها والتحقق منها.

الغرض XslTransform:

تعتبر الحاجة إلى تحويل الوثيقة باستخدام XSL أو XSTL من أهم متطلبات العمل مع وثائق XML.

تقدم NET. الغرض XslTransform المصمم خصيصاً لإجراء عمليات التحويل باستخدام XSL أو XSTL.

الطرق الأساسية لغرض XslTransform:

تقدم XslTrasform طريقتين أساسيتين هما:

الطريقة	الوصف
Load	تقوم بتحميل وثيقة أنماط XSLويتم التأشير إلى أي وثيقة أنماط باستخدام xsl:include
Transform	تقوم بتحويل بيانات وثيقة Xml محددةباستخدام XSL أو XSLTوتقوم بإخراج الناتج.

مثال

نلاحظ في هذا المثال أننا استخدمنا الغرض XmlDocument ثم قمنا بتحميل ملف إليه بالطريقة Load وبعدها قمنا باستخدام الطريقة GetElementByTagname لإيجاد جميع العقد التي يكوناسم التأشيرة فيها هو "AutherName" ثم قمنا بالدوران ضمن النتائج وإظهارها.

```
<%@Page Language="VB" %>
<%@Import Namespace="System.XML" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Searching an XML document using the DOM</title>
<!-- #include file="..\global\style.inc" -->
</head>
<body bgcolor="#ffffff">
<span class="heading">Searching an XML document using the
DOM</span><hr />
<!-----------------------------------------------------------------
---------->

<div id="outDocURL" runat="server"></div>
<div id="outError" runat="server"> </div>
<div id="outResults" runat="server"></div>

<script language="vb" runat="server">

Sub Page_Load()
```

```
'create physical path to booklist.xml sample file (in same folder
as ASPX page)
Dim strCurrentPath As String = Request.PhysicalPath
Dim strXMLPath As String = Left(strCurrentPath,
InStrRev(strCurrentPath, "\")) & "booklist.xml"

'create a new XMLDocument object
Dim objXMLDoc As New XmlDocument()

Try

'load the XML file into the XMLDocument object
objXMLDoc.Load(strXMLPath)
outDocURL.innerHTML = "Loaded file: <b>" & strXMLPath & "</b>"

Catch objError As Exception

'display error details
outError.innerHTML = "<b>* Error while accessing document</b>.<br
/>" _
& objError.Message & "<br />" & objError.Source
Exit Sub  ' and stop execution

End Try

'now ready to parse the XML document
'it must be well-formed to have loaded without error

'create a string to hold the matching values found
Dim strResults As String = "<b>List of authors</b>:<br />"

'create a NodeList collection of all matching child nodes
Dim colElements As XmlNodeList
colElements = objXMLDoc.GetElementsByTagname("AuthorName")

'iterate through the collection getting the values of the
'child #text nodes for each one
Dim objNode As XmlNode
For Each objNode In colElements
strResults += objNode.FirstChild().Value & "<br />"
Next

'then display the result
outResults.innerHTML = strResults   'display the result

End Sub

</script>

</body>
```

```
</html>
```

مثال:

في المثال التالي سنقوم باستخدام غرض XmlDocument ثم تحميل وثيقة XML إليه بواسطة الطريقة Load .

قمنا بتعريف تابع ليقوم بإعادة نوع العقدة ضمن الوثيقة بناءاً على الرقم الذي تعيده الطريقة NodeType .

ثم قمنا بتعريف تابع آخر يقوم باستعراض العقد وفحص العقد الأبناء لكل عقدة.

```
<%@Page Language="VB" %>
<%@Import Namespace="System.XML" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Accessing XML documents using the DOM</title>
<!-- #include file="..\global\style.inc" -->
</head>
<body bgcolor="#ffffff">
<span class="heading">Accessing XML documents using the
DOM</span><hr />
<!-----------------------------------------------------------------
---------->

<div id="outDocURL" runat="server"></div>
<div id="outError" runat="server"> </div>
<div id="outResults" runat="server"></div>

<script language="vb" runat="server">

Sub Page_Load()

'create physical path to booklist.xml sample file (in same folder
as ASPX page)
Dim strCurrentPath As String = Request.PhysicalPath
Dim strXMLPath As String = Left(strCurrentPath,
InStrRev(strCurrentPath, "\")) & "booklist.xml"

'create a new XMLDocument object
Dim objXMLDoc As New XMLDocument()

Try

'load the XML file into the XMLDocument object
objXMLDoc.Load(strXMLPath)
outDocURL.innerHTML = "Loaded file: <b>" & strXMLPath & "</b>"

Catch objError As Exception
```

```
'display error details
outError.innerHTML = "<b>* Error while accessing document</b>.<br
/>" _
& objError.Message & "<br />" & objError.Source
Exit Sub  ' and stop execution

End Try

'now ready to parse the XML document
'it must be well-formed to have loaded without error
'call a recursive function to iterate through all the nodes
'in the document creating a string that is placed in the <div>
above
Dim strNodes As String
outResults.innerHTML = strNodes &
GetChildNodes(objXMLDoc.ChildNodes, 0)

End Sub


Function GetChildNodes(objNodeList As XMLNodeList, intLevel As
Integer) As String

Dim strNodes As String = ""
Dim objNode As XMLNode
Dim objAttr As XMLAttribute

'iterate through all the child nodes for the current node
For Each objNode In objNodeList

'display information about this node
strNodes = strNodes & GetIndent(intLevel) _
& GetNodeType(objNode.NodeType) & ": <b>" & objNode.Name

'if it is an XML Declaration node, display the 'special' properties
If objNode.NodeType = XMLNodeType.XmlDeclaration Then
'cast the XMLNode object to an XmlDeclaration object
Dim objXMLDec =CType(objNode, XmlDeclaration)
strNodes = strNodes & "</b>  version=<b>" & objXMLDec.Version
_
& "</b>  standalone=<b>" & objXMLDec.Standalone & "</b><br />"
Else
'just display the generic 'value' property
strNodes = strNodes & "</b>  value=<b>" & objNode.Value &
"</b><br />"
End If

'if it is an Element node, iterate through the Attributes
'collection displaying information about each attribute
If objNode.NodeType = XMLNodeType.Element Then
```

```
'display the attribute information for each attribute
For Each objAttr In objNode.Attributes
strNodes = strNodes & GetIndent(intLevel + 1) _
& GetNodeType(objAttr.NodeType) & ": <b>" & objAttr.Name _
& "</b>  value=<b>" & objAttr.Value & "</b><br />"
Next
End If

'if this node has child nodes, call the same function recursively
'to display the information for it and each of its child node
If objNode.HasChildNodes Then
strNodes = strNodes & GetChildNodes(objNode.childNodes, intLevel +
1)
End If

Next  'go to next node

Return strNodes   'pass the result back to the caller

End Function


Function GetIndent(intLevel As Integer)
'returns a string of non-breaking spaces used to indent each line
Dim strIndent As String = ""
Dim intIndent As Integer
For intIndent = 0 To intLevel
strIndent = strIndent & "      "
Next
Return strIndent
End Function

Function GetNodeType(intType As Integer) As String
'returns the node type as a string
Select Case (intType)
Case 0: Return "NONE"
Case 1: Return "ELEMENT"
Case 2: Return "ATTRIBUTE"
Case 3: Return "TEXT"
Case 4: Return "CDATA SECTION"
Case 5: Return "ENTITY REFERENCE"
Case 6: Return "ENTITY"
Case 7: Return "PROCESSING INSTRUCTION"
Case 8: Return "COMMENT"
Case 9: Return "DOCUMENT"
Case 10: Return "DOCUMENT TYPE"
Case 11: Return "DOCUMENT FRAGMENT"
Case 12: Return "NOTATION"
Case 13: Return "WHITESPACE"
Case 14: Return "SIGNIFICANT WHITESPACE"
Case 15: Return "END ELEMENT"
```

```
Case 16: Return "END ENTITY"
Case 17: Return "XML DECLARATION"
Case 18: Return "NODE (ALL)"
Case Else: Return "UNKNOWN"
End Select
End Function

</script>

</body>
</html>
```

مثال:

سنستخدم في المثال التالي الغرض XmlDocument مع الغرض XPathNavigator لن نستخدم هنا الغرض XMLNode وسنعيد نفس المثال السابق

```
<%@Page Language="VB" %>
<%@Import Namespace="System.Xml" %>
<%@Import Namespace="System.Xml.Xpath" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Accessing XML documents using an XPathNavigator</title>
<!-- #include file="..\global\style.inc" -->
</head>
<body bgcolor="#ffffff">
<span class="heading">Accessing XML documents using an
XPathNavigator</span><hr />
<!-------------------------------------------------------------------
---------->

<div id="outDocURL" runat="server"></div>
<div id="outError" runat="server"> </div>
<div id="outResults" runat="server"></div>

<script language="vb" runat="server">

Sub Page_Load()

'create physical path to booklist.xml sample file (in same folder
as ASPX page)
Dim strCurrentPath As String = Request.PhysicalPath
Dim strXMLPath As String = Left(strCurrentPath,
InStrRev(strCurrentPath, "\")) & "booklist.xml"

'create a new XmlDocument object
Dim objXMLDoc As New XmlDocument

Try
```

```
'load the XML file
objXMLDoc.Load(strXMLPath)
outDocURL.innerHTML = "Loaded file: <b>" & strXMLPath & "</b>"

Catch objError As Exception

'display error details
outError.innerHTML = "<b>* Error while accessing document</b>.<br
/>" _
& objError.Message & "<br />" & objError.Source
Exit Sub  ' and stop execution

End Try

'now ready to parse the XML document
'it must be well-formed to have loaded without error
'create a new XPathNavigator object using the XMLDocument object
Dim objXPNav As XPathNavigator = objXMLDoc.CreateNavigator()

'move the current position to the root #document node
objXPNav.MoveToRoot()

'call a recursive function to iterate through all the nodes in the
'XPathNavigator, creating a string that is placed in the <div>
above
outResults.innerHTML = GetXMLDocFragment(objXPNav, 0)

End Sub


Function GetXMLDocFragment(objXPNav As XPathNavigator, intLevel As
Integer) As String

Dim strNodes As String = ""
Dim intLoop As Integer

'display information about this node
strNodes = strNodes & GetIndent(intLevel) _
& GetNodeType(objXPNav.NodeType) & ": <b>" & objXPNav.Name _
& "</b>  value=<b>" & objXPNav.Value & "</b><br />"

'see if this node has any Attributes
If objXPNav.HasAttributes Then

'move to the first attribute
objXPNav.MoveToFirstAttribute()

Do

'display the information about it
```

```
strNodes = strNodes & GetIndent(intLevel + 1) _
& GetNodeType(objXPNav.NodeType) & ": <b>" & objXPNav.Name _
& "</b>  value=<b>" & objXPNav.Value & "</b><br />"

Loop While objXPNav.MoveToNextAttribute()

'then move back to the parent node (i.e. the element itself)
objXPNav.MoveToParent()

End If

'see if this node has any child nodes
If objXPNav.HasChildren Then

'move to the first child node of the current node
objXPNav.MoveToFirstChild()

Do
'recursively call this function to display the child node fragment
strNodes = strNodes & GetXMLDocFragment(objXPNav, intLevel + 1)
Loop While objXPNav.MoveToNext()

'move back to the parent node - the node we started from when we
'moved to the first child node - could have used Push and Pop
instead
objXPNav.MoveToParent()

End If

'must repeat the process for the remaining sibling nodes (i.e.nodes
'at the same 'level' as the current node within the XML document
'so repeat while we can move to the next sibling node
Do While objXPNav.MoveToNext()

'recursively call this function to display this sibling node
'and its atributes and child nodes
strNodes = strNodes & GetXMLDocFragment(objXPNav, intLevel)

Loop

Return strNodes  'pass the result back to the caller

End Function


Function GetIndent(intLevel As Integer)
'returns a string of non-breaking spaces used to indent each line
Dim strIndent As String = ""
Dim intIndent As Integer
For intIndent = 0 To intLevel
strIndent = strIndent & "      "
```

```
Next
Return strIndent
End Function

Function GetNodeType(intType As Integer) As String
'returns the node type as a string
Select Case (intType)
Case 0: Return "NONE"
Case 1: Return "ELEMENT"
Case 2: Return "ATTRIBUTE"
Case 3: Return "TEXT"
Case 4: Return "CDATA SECTION"
Case 5: Return "ENTITY REFERENCE"
Case 6: Return "ENTITY"
Case 7: Return "PROCESSING INSTRUCTION"
Case 8: Return "COMMENT"
Case 9: Return "DOCUMENT"
Case 10: Return "DOCUMENT TYPE"
Case 11: Return "DOCUMENT FRAGMENT"
Case 12: Return "NOTATION"
Case 13: Return "WHITESPACE"
Case 14: Return "SIGNIFICANT WHITESPACE"
Case 15: Return "END ELEMENT"
Case 16: Return "END ENTITY"
Case 17: Return "XML DECLARATION"
Case 18: Return "NODE (ALL)"
Case Else: Return "UNKNOWN"
End Select
End Function

</script>

<!-------------------------------------------------------------------
---------->
</body>
</html>
```

سنقوم بعمل مشابه للأمثلة السابقة هذه المرة باستخدام الغرض XmlTextReader

```
<%@Page Language="VB" %>
<%@Import Namespace="System.XML" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Accessing an XML document with an XMLTextReader
object</title>
<!-- #include file="..\global\style.inc" -->
</head>
<body bgcolor="#ffffff">
```

```
<span class="heading">Accessing an XML document with an
XMLTextReader object</span><hr />
<!-------------------------------------------------------------------
---------->

<div id="outDocURL" runat="server"></div>
<div id="outError" runat="server"> </div>
<div id="outResults" runat="server"></div>

<script language="vb" runat="server">

Sub Page_Load()

'create physical path to booklist.xml sample file (in same folder
as ASPX page)
Dim strCurrentPath As String = Request.PhysicalPath
Dim strXMLPath As String = Left(strCurrentPath,
InStrRev(strCurrentPath, "\")) & "booklist.xml"

'declare a variable to hold an XmlTextReader object
Dim objXMLReader As XmlTextReader

Try

'create a new XmlTextReader object for the XML file
objXMLReader = New XmlTextReader(strXMLPath)
outDocURL.innerHTML = "Opened file: <b>" & strXMLPath & "</b>"

Catch objError As Exception

'display error details
outError.innerHTML = "<b>* Error while accessing document</b>.<br
/>" _
& objError.Message & "<br />" & objError.Source
Exit Sub  ' and stop execution

End Try

'now ready to read (or "pull") the nodes of the XML document
Dim strNodeResult As String = ""
Dim objNodeType As XmlNodeType

'read each node in turn - returns False if no more nodes to read
Do While objXMLReader.Read()

'select on the type of the node (these are only some of the types)
objNodeType = objXMLReader.NodeType

Select Case objNodeType

Case XmlNodeType.XmlDeclaration:
```

```
'get the name and value
strNodeResult += "XML Declaration: <b>" & objXMLReader.Name _
& " " & objXMLReader.Value & "</b><br />"

Case XmlNodeType.Element:
'just get the name, any value will be in next (#text) node
strNodeResult += "Element: <b>" & objXMLReader.Name & "</b><br />"

Case XmlNodeType.Text:
'just display the value, node name is "#text" in this case
strNodeResult += "  - Value: <b>" & objXMLReader.Value _
& "</b><br />"

End Select

'see if this node has any attributes
If objXMLReader.AttributeCount > 0 Then

'iterate through the attributes by moving to the next one
'could use MoveToFirstAttribute but MoveToNextAttribute does
'the same when the current node is an element-type node
Do While objXMLReader.MoveToNextAttribute()

'get the attribute name and value
strNodeResult +=  "  - Attribute: <b>" & objXMLReader.Name _
& "</b>   Value: <b>" & objXMLReader.Value _
& "</b><br />"
Loop

End If

Loop   'and read the next node

'finished with the reader so close it
objXMLReader.Close()

'and display the results in the page
outResults.innerHTML = strNodeResult

End Sub

</script>
<!-------------------------------------------------------------------
---------->
</body>
</html>
```

مثال:

في المثال التالي سنقوم باستخدام الغرض XMLTextWriter وسنقوم بكتابة وثيقة XMl بواسطته. نلاحظ

أننا بدأنا بإنشاء الغرض XmlTextWriter ثم قمنا بتحديد التنسيق (مقدار الإزاحة بين العناصر) من ثم بدأنا بكتابة الوثيقة باستخدام طرق الغرض مثل WriteStartDocument و WriteComment و WriteStartElement....

ثم قمنا بتفريغ محتوى الغرض إلى الملف وإغلقنا الملف باستخدام الطريقتين flush و Close.

بعدها قمنا بفتح الملف وتحميل محتواه إلى سلسة محارف ثم قمنا بكتابة المحتوى إلى عنصر التحكم من جهة المخدمOurResult

```
<%@Page Language="VB" %>
<%@Import Namespace="System.XML" %>
<%@ Import Namespace="System.IO" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Creating an XML document with an XMLTextWriter
object</title>
<!-- #include file="..\global\style.inc" -->
</head>
<body bgcolor="#ffffff">
<span class="heading">Creating an XML document with an
XMLTextWriter object</span><hr />
<!-----------------------------------------------------------------
---------->

<div id="outDocURL" runat="server"></div>
<div id="outError" runat="server"> </div>
<div id="outResults" runat="server"></div>

<script language="vb" runat="server">

Sub Page_Load()

'create physical path for the new file (in same folder as ASPX
page)
Dim strCurrentPath As String = Request.PhysicalPath
Dim strXMLPath As String = Left(strCurrentPath,
InStrRev(strCurrentPath, "\")) & "newbooklist.xml"

'declare a variable to hold an XmlTextWriter object
Dim objXMLWriter As XmlTextWriter

Try

'create a new objXMLWriter object for the XML file
objXMLWriter = New XmlTextWriter(strXMLPath, Nothing)
outDocURL.innerHTML = "Writing to file: <b>" & strXMLPath & "</b>"
```

```
Catch objError As Exception

'display error details
outError.innerHTML = "<b>* Error while accessing document</b>.<br
/>" _
& objError.Message & "<br />" & objError.Source
Exit Sub  ' and stop execution

End Try

'now ready to write (or "push") the nodes for the new XML document

'turn on indented formatting and set indent to 3 chararcters
objXMLWriter.Formatting = Formatting.Indented
objXMLWriter.Indentation = 3

'start the document with the XML declaration tag
objXMLWriter.WriteStartDocument()

'write a comment element including the current date/time
objXMLWriter.WriteComment("Created using an XMLTextWriter - " &
Now())

'write the opening tag for the <BookList> root element
objXMLWriter.WriteStartElement("BookList")

'write the opening tag for a <Book> element
objXMLWriter.WriteStartElement("Book")

'add two attributes to this element's opening tag
objXMLWriter.WriteAttributeString("Category", "Technology")
Dim intPageCount As Integer = 1248   'numeric value to convert
objXMLWriter.WriteAttributeString("Pagecount",
intPageCount.ToString("G"))

'write four elements, using different source data types
objXMLWriter.WriteElementString("Title", "Professional Video
Recorder Programming")
Dim datReleaseDate As DateTime = #03/03/2000#
objXMLWriter.WriteElementString("ReleaseDate",
datReleaseDate.ToString("yyyy-MM-dd"))
Dim intSales As Integer = 17492
objXMLWriter.WriteElementString("Sales", intSales.ToString("G"))
Dim blnHardback As Boolean = True
objXMLWriter.WriteElementString("Hardback", blnHardback.ToString())

'write the opening tag for the <AuthorList> child element
objXMLWriter.WriteStartElement("AuthorList")

'add two <Author> elements
objXMLWriter.WriteElementString("Author", "Francesca Unix")
```

```
objXMLWriter.WriteElementString("Author", "William Soft")

'close the <AuthorList> element
objXMLWriter.WriteEndElement()

'close the <Book> element
objXMLWriter.WriteEndElement()

'close the root <BookList> element
objXMLWriter.WriteEndElement()

'flush the current content to the file and close it
objXMLWriter.Flush()
objXMLWriter.Close()

'now open the new XML file and read it into a string
Dim strXMLResult As String
Dim objSR As StreamReader = File.OpenText(strXMLPath)
strXMLResult = objSR.ReadToEnd()
objSR.Close
objSR = Nothing

'and display the results in the page
outResults.innerHTML = "<pre>" & Server.HtmlEncode(strXMLResult) &
"<pre>"

End Sub
</script>
<!-----------------------------------------------------------------
---------->

</body>
</html>
```

الفصل الخامس عشر والسادس عشر

عنوان الموضوع:

الأمان في ASP.NET

الكلمات المفتاحية:

التحقق من الهوية ، السماحية، مستخدم، مجموعة، كعكة، جلسة.

ملخص:

يعد موضوع الأمان موضوع ذو حساسية كبيرة بالأخص في بيئة كبيئة الوب. سنغطي خلال هاتين الجلستين مواضيع الأمان بما يشمل ASP.NET وتداخلاتها مع الأمان في IIS و Windows

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- مفاهيم الأمان
- التحقق من الهوية في IIS و Windows
- التحكم بالسماحيات في IIS و Windows
- إجراءات الأمان الخاصة ب ASP.NET

الأمان في ASP.NET

جرى تصميم أغلب الصفحات على الانترنت ليتم الوصول إليها من أي زائر. لذا تكون الإعدادات التلقائية في ASP.NET أمثلية، فأي شخص يستطيع الوصول إلى أي صفحة من أي مكان على الانترنت.

لكننا نحتاج دائماً إلى جعل صفحات غير متوفرة للعامة. فعلى سبيل المثال قد تحتاج إلى حصر الوصول إلى موقع كامل للمستخدمين غير المسجلين، أو حصر الوصول إلى صفحات معينة لمدير النظام فقط.

عموماً، كانت عملية تأمين الصفحات في النسخة السابقة (ASP) تتم باستخدام إحدى طريقيتن :

- إنشاء نظام أمان مخصص يسمح للمستخدمين بالدخول إلى الموقع أو التطبيق
- الاعتماد على خصائص الأمان في IIS و Windows لتحديد المستخدمين الذين يمكن لهم الوصول إلى الصفحات، أو المجلدات، أو الموارد

تكون الصفحات في ASP.NET عاملة في إطار NET. حيث تحافظ على ميزات الأمان القديمة مع إضافة مفاهيم جديدة سنستعرضها في الشرائح القادمة، كما سنستعرض جميع العمليات التي تتحكم في وصول المستخدم ونركز على تلك المصممة للعمل خصيصاً مع ASP.NET.

مفاهيم الأمن

يرتكز الأمن عادةً على أربعة مواضيع رئيسية:

- التحقق من الهوية
- الصلاحيات والسماحيات
- التمثيل
- الأمان الوظيفي أو أمان البيانات

يرتكز الأمن عادةً على أربعة مواضيع رئيسية:

التحقق من الهوية:

وهي العملية التي نقوم بها لتمييز هوية كل مستخدم بجعله يثبت أنه من يدّعي. فعند السماح لمستخدم معين بالوصول إلى موارد منظومة، يجب أن نكون قادرين على التعرف عليه. ويجري الأمر غالباً باستخدام اسم دخول وكلمة سر أو شهادة رقمية أو بطاقة ذكية أو حتى قارئ بصمات.

الصلاحيات والسماحيات:

بعد معرفة من هو المستخدم نستطيع اتخاذ قرار بشأن إمكانياته وسماحيات وصوله للموارد المختلفة. تجري هذه العملية باستخدام مجموعة من الطرق كقائمة التحقق من الوصول إلى غرض (ACL). فعلى سبيل المثال: إذا طلب أحد الأشخاص الوصول إلى صفحة ASP، يتحقق النظام من كون هذا الشخص يملك حق قراءة هذه الصفحة قبل أن يسمح لـــ IIS بجلب تلك الصفحة.

التمثيل:

يُعرَّف التمثيل على أنه وصول لمصدر ما باسم آخر وتحت هوية شخص آخر. فعلى سبيل المثال، تُستخدم آلية التحقق من الهوية مع جميع الصفحات حتى تلك المفتوحة على الجميع، ولكن الأمر يجري في الحالة الأخيرة باستخدام حسابين لهما طابع الضيف (Guest) أو المجهول (Anonymous) يكون الحساب الأول باسم IUSER_%machinename% و الحساب الثاني IWAM_%machinename%.

يجري تجهيز هذه الحسابات عند تنصيب IIS، وتجري إضافتها آلياً إلى جميع المجلدات في جميع المواقع على المخدم. فإذا سمحنا بوصول غير مشروط إلى مصدر ما في IIS سيبدو جميع المستخدمين كمستخدم واحد من خلال الحساب IUSER_%machinename% الذي سيسمح بالوصول إلى المصدر بالنيابة عن المستخدم.

أما إذا كان المصدر المراد الوصول إليه صفحة تستخدم المكونات COM أو COM+ فيجري استخدام

الحساب IWAN_machinename لينوب عن المستخدم غير المعرَّف.

يقتصر التحكم بالوصول إلى الملفات في ASP.NET على تلك المعرّفة كملفات تطبيق وهي تشمل aspx. ، ax. ، cx. ، vb. ، cs. إضافة إلى ملفات خدمات الوب وموارد أخرى تم ربطها إلى الملف ASPNET_isapi.dll . ولاتطبق قواعد الوصول إلى ملفات مثل الصور وملفات doc، zip، pdf حيث يجري التحكم بحماية هذه الملفات باستخدام تقنيات Windows المعيارية مثل ACL.

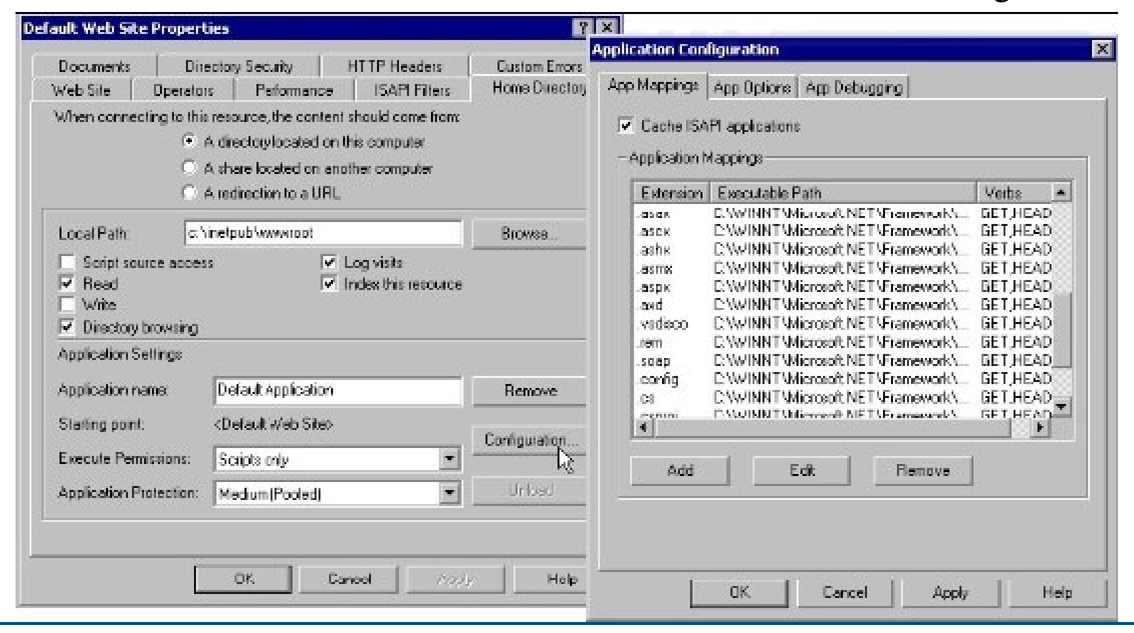
الأمان الوظيفي أو أمان البيانات:

يتضمن عملية تأمين النظام بالمعنى الفيزيائي كتحديث نظام التشغيل واستخدام نسخ مستقرة من البرامج.

الأمان في ASP.NET

ذكرنا سابقاً بعض الفروقات بين الطريقة التقليدية التي كانت متبعة لضبط الإعدادات في ASP.NET وبين سلفها ASP، ورأينا أن عملية الضبط تتم باستخدام ملف web.config الذي يقوم بتعديل المعلومات التينفسها التي يمكن إعدادها من واجهة مدير خدمات الإنترنت ISM (ضمن MMC) .

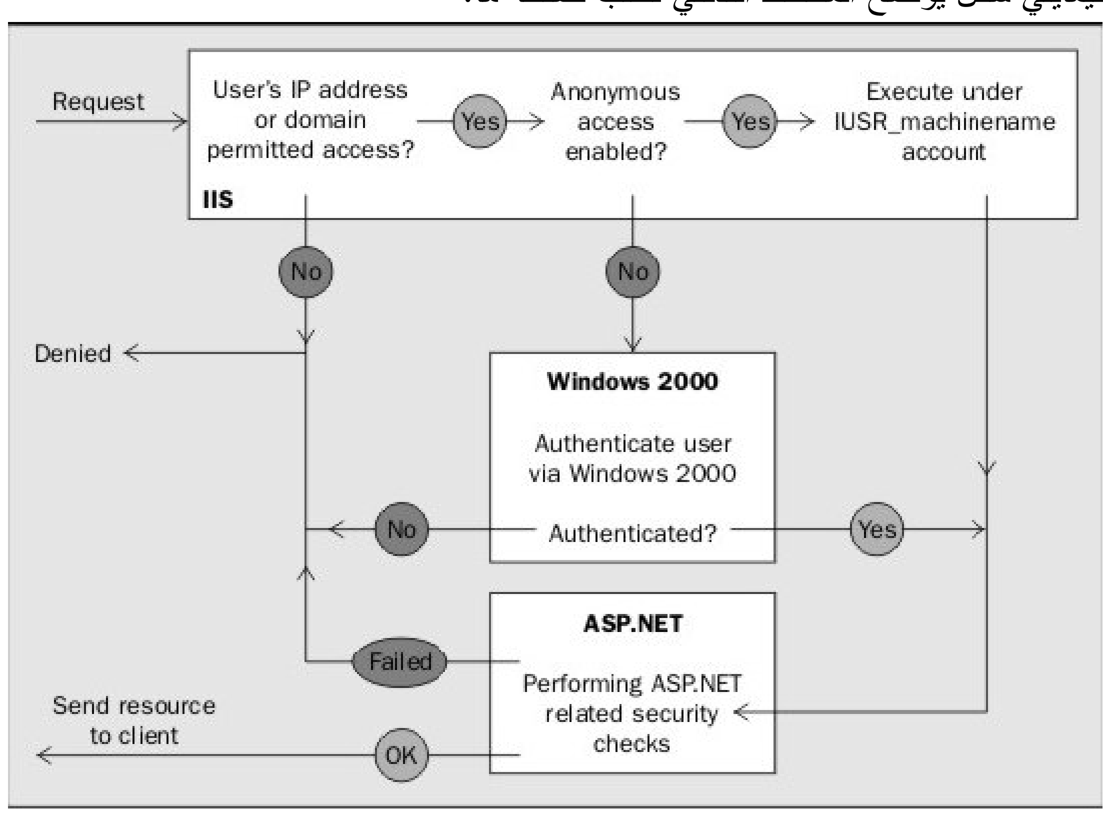
على أي حال، تبقى إعدادات الأمان المطبقة في IIS فعالة لأنه –وبعكس الإعدادات الخاصة بالتطبيقات– ما يزال IIS يدير الطلبات وعمليات الأمان بالتعاون مع ASP.NET إذ يتسلم IIS الطلب ويقوم بتحويله إلى ASP.NET . يمكننا رؤية الإعدادات المتعلقة بعملية التحويل تلك من خلال فتح نافذة Application Configuration :



نلاحظ أنه يتم تحويل جميع الطلبات التي تتناول اللاحقات الظاهرة إلى الملف aspnet_isapi.dll في المجلد الخاص بإطار عمل NET.



فيمايلي شكل يوضح المخطط التدفقي لطلب صفحة ما:



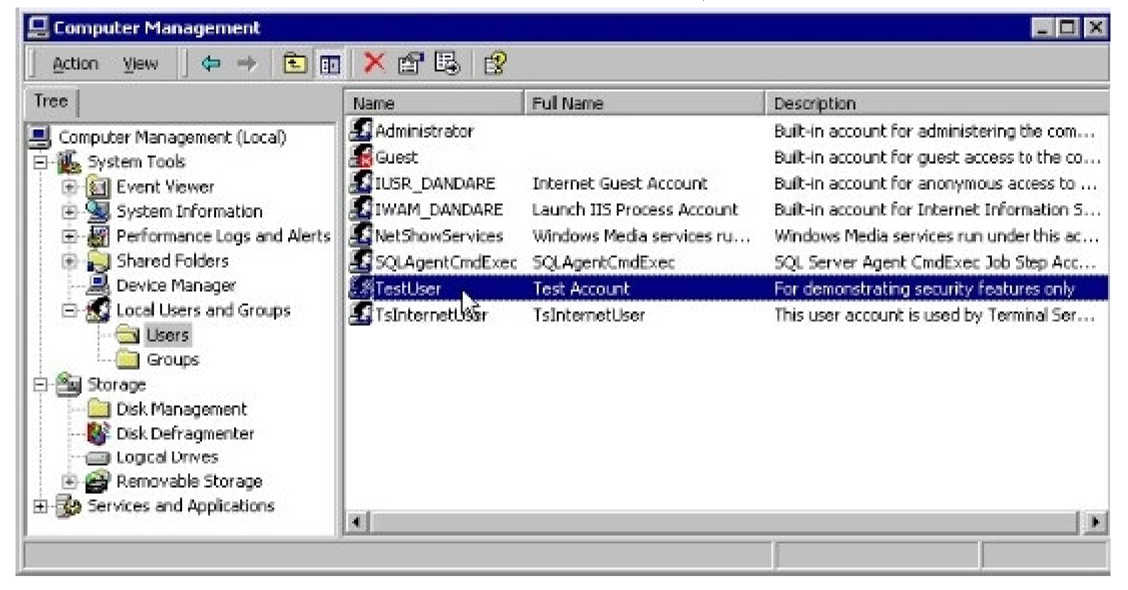
ترجمة الشكل	
Request	طلب
User's IP Address or domain Permitted access?	هل عنوان المستخدم أو عنوان المجال مقبول؟
Anonymous Access Enabled?	هل يُسمح بالدخول المجهول الهوية؟

Execute under IUSR_%machinename% account	تشغيل تحت حساب IUSR_%machinename%
Denied	ممنوع
Send Resource to Client	إرسال المورد إلى الزبون
Authenticate user via Windows 2000	التحقق من هوية المستخدم اعتماداً على Windows 2000
Authenticated	جرى التحقق من الهوية
Performing ASP.Net Related Security checks	تنفيذ إجراءات الأمان الخاصة بـــ ASP.NET
Yes	نعم
No	كلا

الصلاحيات في Windows2000

تحتفظ Windows 2000 بقائمة من المستخدمين الذين يُسمح لهم الوصول إلى المصادر المختلفة على الجهاز. يمكن أن تكون هذه القائمة محفوظة على الجهاز نفسه أو على جهاز آخر يعمل كتحكم نطاق.

يجري التحكم بهذه القائمة عن طريق أداة ComputerManagement أو من خلال أداة Active Directory Users ضمن أدوات التحكم بالنطاق.



مجموعات المستخدمين:

توفر مجموعات المستخدمين العناء في حال رغبنا بتوزيع مجموعة كبيرة من المستخدمين بصلاحيات متشابهة. فعوضاً عن إعداد صلاحيات 500 مستخدم للوصول إلى مورد ما والاضطرار إلى تغيير هذه الصلاحيات بصورة إفرادية عند طلب أي تغيير. يكفي إنشاء مجموعة وإعطاء هذه المجموعة الصلاحيات المطلوبة على المصدر المحدد. وفي حال طرأ أي تغيير على تلك الصلاحيات يكفي تغيير الصلاحيات الخاصة بالمجموعة لتؤثر على كل المستخدمين الذين ينتمون إليها.
توضح النافذة التالية مستخدم عضو في مجموعتين GroupTest و Users

التحقق من الهوية في IIS

عندما يقوم مستخدم بطلب مورد ما باستخدام الوب، يقوم IIS باستلام الطلب وإجراء عملية التحقق الأساسية عليه، كما يقوم بمجموعة من الاختبارات قبل منحه الوصول إلى المصدر المطلوب :

اختبار عنوان IP و النطاق:
يمكننا في Windows 2000 Server و NT تخصيص عناوين IP أو أسماء نطاق للزبائن المسموح لهم الوصول إلى الموارد المختلفة. يتم ذلك باستخدام واجهة IP Address and Domain Name من التبويب Directory Security في النافذة Properties الخاصة بموقع ما ضمن IIS.
تساعد هذه العملية في الوصول بصورة دائمة إلى الموقع من عنوان أو مجموعة محددة من العناوين.

استخدام الشهادات الرقمية لتعريف المستخدمين:
يمكننا أيضاً استخدام النافذة Properties لإعداد مخدم شهادات رقمية ليجري استخدامها من أجل موقع ما. كذلك يمكننا تفعيل الاتصال الآمن باستخدام SSL. يمكن لهذه الشهادات الرقمية أن تُستخدم مع الشهادات من طرف الزبون للتعرف على الجهاز الذي يحاول الوصول إلى المخدم.

تحديد آلية التحقق من الهوية في IIS

نلاحظ في النافذة الخاصة ب Directory Security وجود خيار يسمح لنا بتحديد طريقة التحقق من الهوية التي يجب استخدامها.

تقدم نافذة Authentication Methods أربعة خيارات رئيسية:

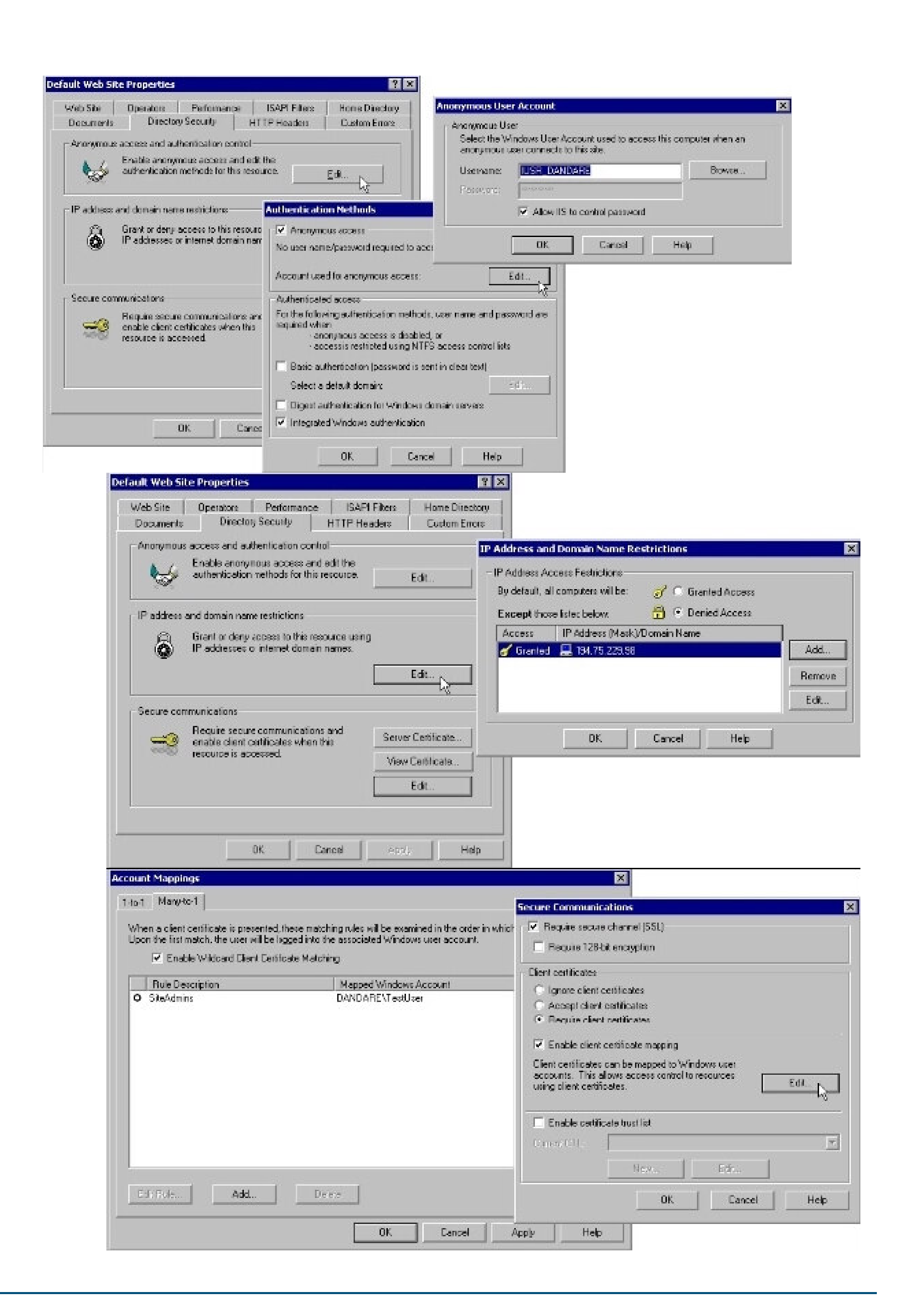
الوصول بدون اسم ، حيث يمكن لأي مستخدم الوصول إلى خدمة WWW في حال لم يجر حجبه بحسب عنوان IP الخاص به أو بسبب القيد الخاص باسم النطاق. يقوم IIS عندها بالوصول بالنيابة عن المستخدم باستعمال الحساب IUSER أو IWAM.

الوصول باستخدام اختبار الهوية الأساسي: عند إلغاء تفعيل الوصول بدون اسم. يجري توليد نافذة تسجيل دخول للمستخدمين من قبل المستعرض. حيث يتم تشفير اسم المستخدم وكلمة السر وإرسالها إلى IIS . عندها يقوم IIS بالبحث عن الحساب في Windows ويمنح سماحية الوصول للمستخدم على المصادر التي يمتلك أحقية الوصول إليها. لا بد من ملاحظة أن التشفير المستخدم والمبني على 64bit ليس آمناً بقدر كبير لذلك لا يعتبر هذا الخيار مناسباً في التطبيقات التي تحتاج درجة أمان عالية.

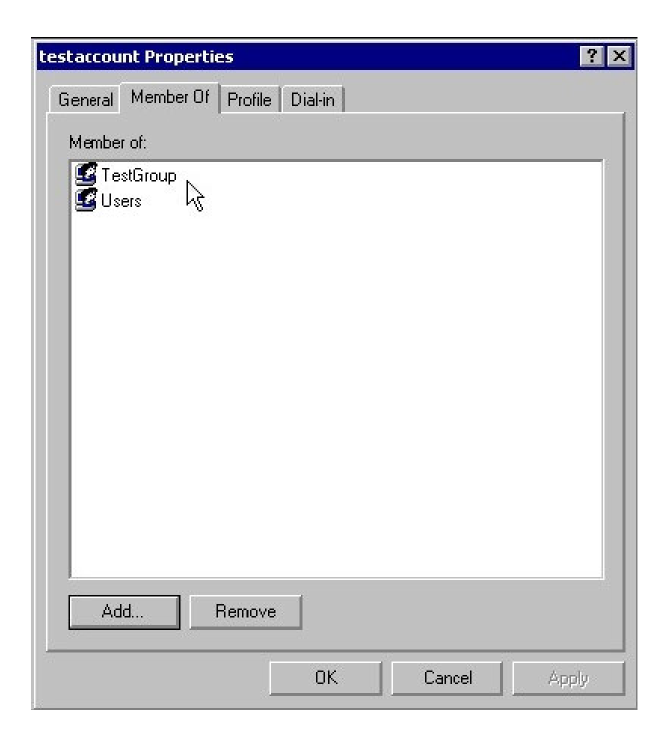
التحقق من الهوية عن طريق إرسال مجموعة من المعلومات المشفرة إلى المخدم (Digest): إذا تم إلغاء تفعيل الوصول بدون اسم، يجري إظهار نافذة تطلب معلومات اسم المستخدم وكلمة السر. يقوم المستعرض بجمع هذه المعلومات مع معلومات أخرى مخزنة على الجهاز الزبون وإرسالها بعد تشفيرها إلى المخدم (باستخدام توابع تشفير باتجاه واحد). بالطبع، يملك المخدم نسخته من هذه المعلومات ويقوم بتشفيرها بدوره ثم تتم المقارنة بين ما وصل إلى المخدم والنسخة التي تم إنشاؤها. تعمل هذه الطريقة فقط باستخدام مستعرض الانترنت IE وخدمات انترنت NET. ولكنها فعالة حتى أن بإمكانها تخطي حواجز كالمخدمات الوكيلة أو جدران النار، وهي آمنة إذ لن يُسمح للمستخدم بالوصول إلى مصدر ما مالم يملك هذا المستخدم حساب صالح على Windows ويكون لهذا الحساب صلاحيات الوصول إلى المورد المحدد.

استخدام تحقق الهوية الخاص بـــ Windows: تستخدم Windows عادة طريقة مشابهة للطريقة السابقة وذلك باعتماد برتوكول تشفير مثل Kerberos أو استخدام البروتوكول NTLM الفرق الأساسي في أن هذه الطريقة غير ملائمة للعمل على الانترنت لأن هذه العملية لا يمكن أن تتم من خلف جدار نار أو ملقم وكيل.
في حال تم إلغاء التفعيل للدخول بدون اسم وتم تفعيل كل الخيارات الأخرى، سيستعمل IIS تحقق الهوية الخاص بـــ Windows ثم تحقق الهوية بالطريقة Digest ثم تحقق الهوية الأساسي كحل أخير في حال كون الزبون لا يدعم الطريقتين الأولى والثانية.

يمكننا استخدام النافذة الخاصة ب Authentication Methods لتحديد الحساب المراد استخدامه في حالة الدخول بدون اسم

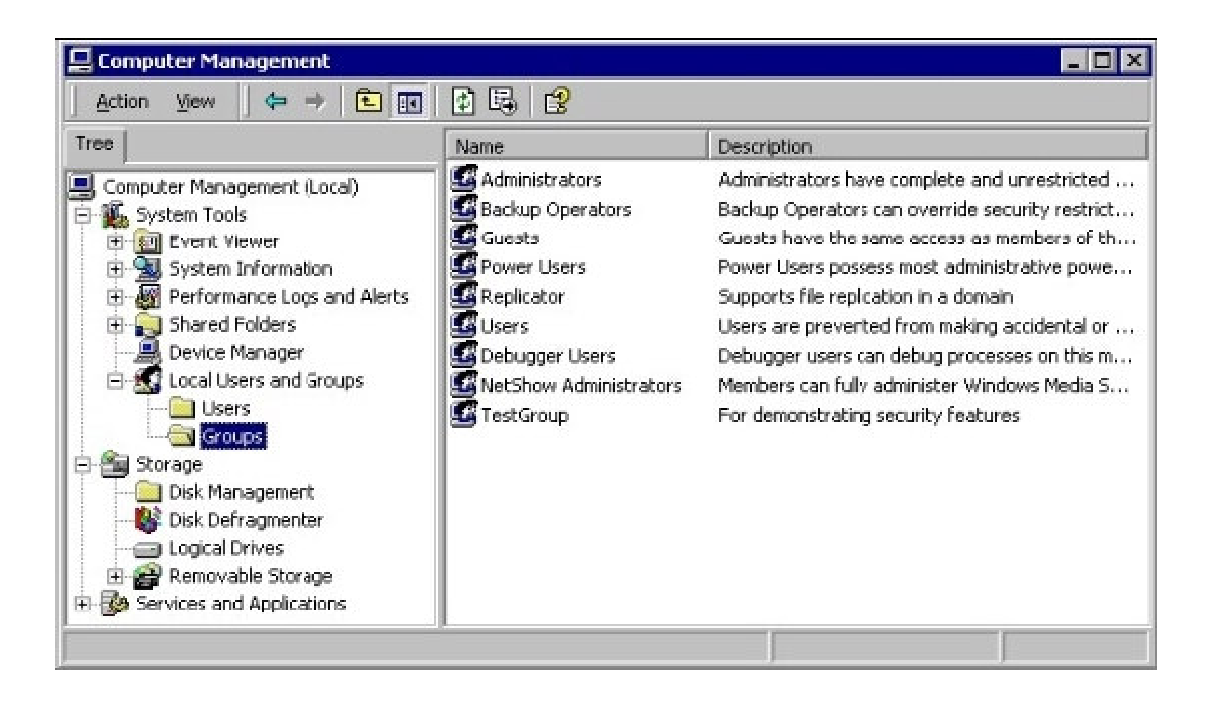


يمكن الوصول إلى المحتوى نفسه برمجياً باستخدام Request.ClientCertificateCollection.



تجري إضافة جميع الحسابات آلياً إلى إلى مجموعة Users.

نلاحظ في الشكل التالي جميع المجموعات المتوفرة تلقائياً إضافة إلى المجموعة الجديدة TestGroup التي قمنا بإنشائها.



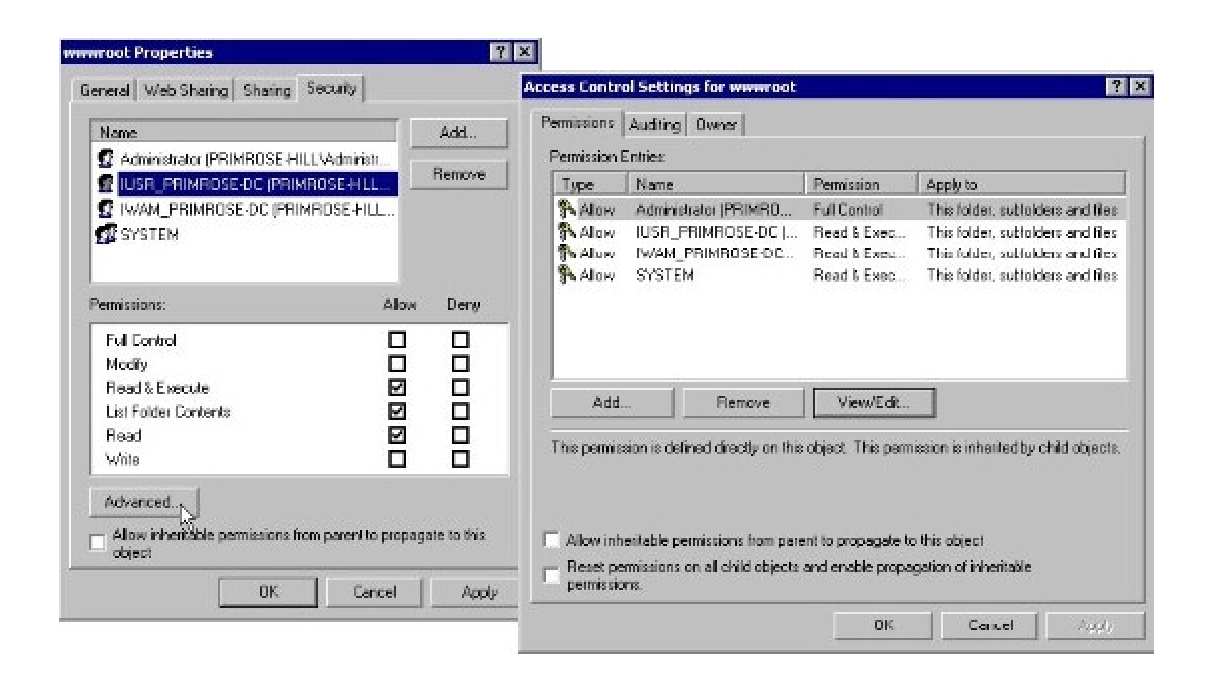
السماحيات في Windows 2000

بعد التحقق من هوية المستخدمين، لن يتمكن أي منهم من الوصول إلى المصادر التي لا يمتلك سماحية الوصول إليها. يتم تخزين معلومات السماحيات عادةً، في قائمة تحكم الوصول ACL المخصصة لكل مصدر من المصادر.

يمكن إدارة هذه القائمة بعدة طرق: فمثلاً يمكن باستخدام إدارة السماحيات على الملفات والمجلدات المحلية والشبكية.

من نافذة Property نختار التبويب Security مما يسمح بعرض الحسابات والمجموعات والسماحيات التي تملكها كل منها في الوصول إلى ملف أو مجلد.

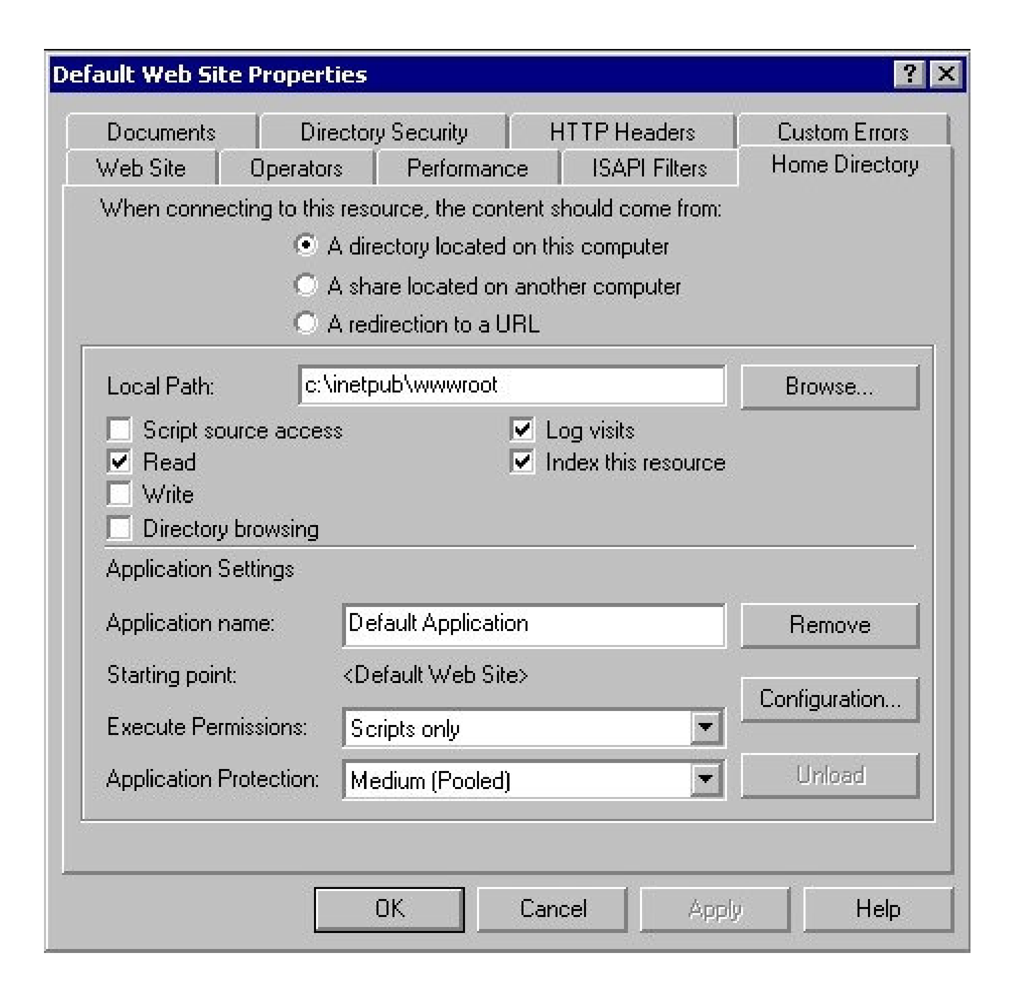
يمكن باستخدام الزر Advanced إظهار نافذة للتحكم بتفاصيل أكثر معطية المجال ل 13 تركيب مختلف من صلاحيات القراءة/الكتابة/الحذف.مع إمكانية التحكم بتوريث المجلدات الأبناء السماحيات المطبقة على المجلدات الأباء.



السماحيات في IIS

يتدخل IIS أيضاً في تحديد سماحيات الوصول على الموارد كونه يقوم بالوصول إلى الموارد بالنيابة عن المستخدم باستعمال الحساب (IUSER) أو أي حساب آخر يحدده المستخدم، كما يحدد على سوية أخرى ما الذي يمكن أن يفعل المستخدم بالمصدر الذي تم الوصول إليه.

يظهر في الجزء الوسطي من تبويب Home Directory الظاهر في الشكل أدناه العمليات الممكنة على الموقع أو المجلد قراءة/كتابة/تشغيل النصوص البرمجية .

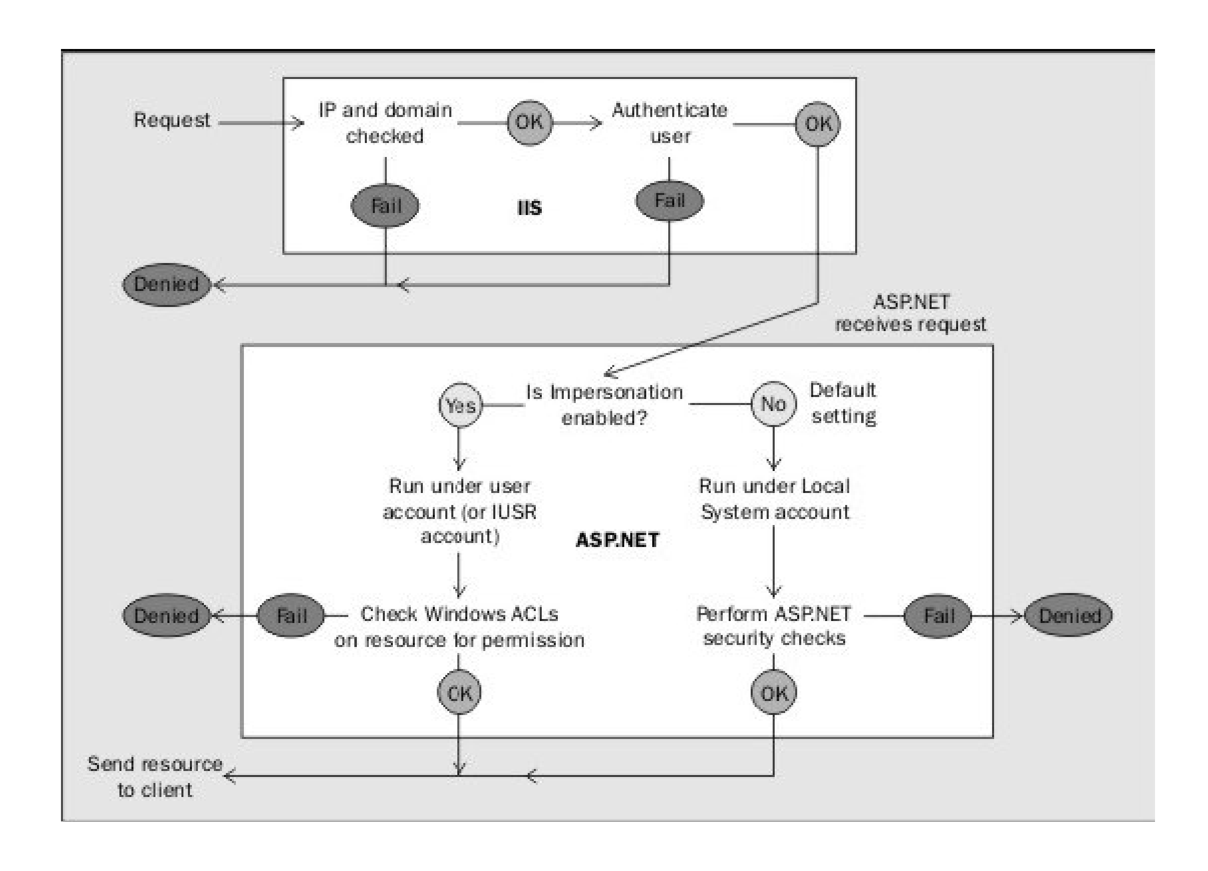


تكون هذه السماحيات منفصلة عن تلك التي يطبقها Windows ويتم تطبيقها اعتماداً على المجلد أو الموقع مما يوفر طبقة حماية إضافية، إذ تعمل السماحيات التي يحددها Windows مع سماحيات IIS.

إجراءات الأمان في ASP.NET

بعد استعراضنا لمجموعة من الأساسيات المتعلقة بأمان نظام التشغيل وأمان IIS سنرى كيف ترتبط هذه المواضيع بميزات الأمان المتوفرة في ASP.NET.

يوضح الشكل التالي إجراءات الأمان في ASP.NET بتفصيل أكبر.



ترجمة الشكل	
Request	طلب
IP and Domain checked	والنطاقIPتم اختبار العنوان
Authenticate User	التحقق من هوية المستخدم
ASP.NET receives request	الطلبASP.NETتستلم
Is impersonation enabled?	هل تم السماح بالتمثيل؟
Run under user account (or IUser account)	العمل تحت اسم حساب مستخدم (أو تحت IUSER(اسم الحساب
Check windows ACLs on resource for permission	التحقق من السماحيات في قوائم تحكم على الموردWindowsوصول
Run under local system account	العمل باستخدام حساب محلي خاص بالنظام
Performing ASP.Net Related Security checks	تنفيذ إجراءات الأمان الخاصة بـــ ASP.NET
Denied	ممنوع
Send Resource to Client	إرسال المورد إلى الزبون
Yes	نعم

No	كلا

إجراءات الأمان في ASP.NET

التمثيل في ASP.NET:

تتلخص الخطوة الأولى ضمن ASP.NET في اختبار فيما إذا كان التطبيق مضبوطاً لاستخدام التمثيل.
يكون المفهوم مشابه لمفهوم التمثيل في IIS مع ضرورة اتخاذ القرار فيما إذا كان طلب المستخدم سينفذ بصلاحيات المستخدم أم بصلاحيات حساب خاص تستخدمه ASP.NET للطلبات بدون اسم. يمكن التحكم بحساب المستخدم فعلياً من خلال العنصر <ProcessModel> ضمن ملف الإعدادات machine.config.
يكون اسم المستخدم التلقائي هو "machine" وتكون كلمة السر "AutoGenerate".

- إذا كان التمثيل مفعلاً في ASP.NET
  - إذا كان الدخول من دون اسم مفعلاً في IIS يتم التعامل مع الطلب باستخدام الحساب الخاص بالدخول بدون اسم في IIS وهو IUSER_%machinename% .
  - إذا كان الوصول بدون اسم غير مفعل في IIS عندها يتم التعامل مع الطلب اعتماداً على حساب Windows الخاص بالمستخدم مرسل الطلب.
  - في كلا الحالتين يتم التحقق من قائمة ACL للموارد التي تم طلبها من قبل المستخدم حيث يجري السماح بالوصول إلى المورد إذا كان الحساب المستخدم يسمح بالعمل مع هذا المورد.

- إذا كان التمثيل غير مفعل في ASP.NET:
  - إذا كان الدخول بدون اسم مفعل يتم التعامل مع الطلب باستخدام الحساب الخاص بالإجراء ASP.NET
  - إذا كان الدخول بدون اسم غير مفعل يتم التعامل مع الطلب اعتماداً على حساب Windows الخاص بالمستخدم مرسل الطلب.
  - في كلا الحالتين يتم التحقق من قائمة ACL للموارد التي تم طلبها من قبل المستخدم حيث يجري السماح بالوصول إلى المورد إذا كان الحساب المستخدم يسمح بالعمل مع هذا المصدر.

خيارات الأمان في ASP.NET

تقدم ASP.NET مجموعة من الخيارات لتطبيق الأمان وتقييد وصول المستخدم في تطبيقات الوب .يمكن ضبط كل هذه الخيارات ضمن الملف web.config الموجود ضمن المجلد الجذر للتطبيق.

قمنا سابقاً بدراسة كيفية استخدام هذا الملف لذلك سنركز على مواضيع التعامل مع التحقق من الهوية والسماحيات.

أنواع تحقق الهوية والسماحيات:

تقدم Asp.Net ثلاثة أنواع من تحقق الهوية والسماحيات. كما يمكننا الاعتماد فقط على IIS ليقوم بكامل العمل. الخيارات هي :

النوع	الاسم	الوصف
تحقق الهوية الخاص بWindows	Windows	يتم التحقق من الهوية من قبل IIS باستخدام الطرق المذكورة سابقاً (Basic,Digest,Windows)
تحقق الهوية المبني علىPassport	Passport	يستخدم هذا الخيار طريقة التحقق من الهوية المركزية التي تقدمها شركة مايكروسوفت.
التحقق من الهوية على مستوى النماذج	Forms	يتم توجيه الطلبات التي لم يتم التحقق من هوية مرسليها إلى نماذج HTML. هذه الطريقة مشابهة لتلك المستخدمة سابقاً في ASPولكن يتم تقديم أغلب الوظائف هنا كجزء من إطار عمل NET. يُدخل المستخدم معلومات اسم المستخدم وكلمة المرور. يقوم التطبيق بقبول هوية المستخدم وإنشاء كعكة تحتوي معلوماته . يتم إرسال الطلبات اللاحقة مرفقة مع معلومات الكعكة بحيث تضمن استمرار تعرف صفحات التطبيق على هذا المستخدم.
التحقق من الهوية	None	هذه هي الوضعية التلقائية ، يمكن أن

IISالتلقائي في		يستخدم التمثيل أيضاً ولكن التحكم بالوصول يبقى مرتبطاً بالقيود المحددة في IIS. يتم الوصول إلى المصادر باستخدام حساب ASPNET الخاص بالإجراء أو بواسطة الحساب IUSER في حال تفعيل التمثيل.

خيارات الأمان في ASP.NET

يتم تحديد نوع التحقق من الهوية المطلوب لتطبيق أو مجلد ضمن العنصر <authentication> في ملف web.config لهذا الموقع أو المجلد كما يلي:

```
<configuration>
...
<system.web>
<authentication mode="Windows|Passport|Forms|None">
authentication options used for the application
</authentication>
<authorization>
users and roles that have access to the application
</authorization>
<identity>
if application should run under a different account
</identity>
</system.web>
...
</configuration>
```

العنصران الآخران اللذان نهتم بهما ضمن القسم <system.web> من الملف web.config هما <authorization> و<identity>. يُستخدم الأول لتحديد المستخدمين والمجموعات التي تستطيع أو لا تستطيع الوصول إلى التطبيق. ويُستَخدَم الثاني لتحديد فيما إذا كان التمثيل مفعل أم لا.

خيارات الأمان في ASP.NET

استخدام التحقق من الصحة من النوع Windows :

يناسب هذا النوع تطبيقات إنترانت أو التطبيقات التي يمكننا مسبقاً تحديد المستخدمين الذين يحتاجون للوصول إليها وذلك لضرورة إعداد حساب Windows للمستخدم الراغب بالعمل على هذه التطبيقات.

يتم إعداد استخدام هذا النوع ضمن الملف web.config على الشكل:

```
<configuration>
...
<system.web>
<authentication mode="Windows" />
<identity impersonate="true" />
</system.web>
</configuration>
```

تحديد المستخدمين والمجموعات:

يمكننا تحديد قائمة بالمستخدمين والمجموعات التي لها الحق بالوصول إلى التطبيق وذلك ضمن العنصر <authorization> في سلسة من عناصر <allow> و<deny> وذلك بالشكل:

```
<allow roles="comma-separated list of Windows account group names"
users="comma-separated list of Windows user account names"
verb="GET|POST|HEAD"
/>
<deny roles="comma-separated list of Windows account group names"
users="comma-separated list of Windows user account names"
verb="GET|POST|HEAD"
/>
```

هناك أيضاً رمزان يمكن استخدامهما .

الرمز (*) للتعبير عن جميع المستخدمين أو المجموعات أو الأفعال.

الرمز (؟) للتعبير عن الدخول بدون اسم وفي هذه الحالة يتم إعداد IIS لاستخدام الدخول بدون اسم (لا يمكن استخدام هذا الرمز إلا في الواصفة Users.)

تتم معالجة سلسة عناصر <allow> و<deny> من الأعلى إلى الأسفل ويستمر حتى في حال إيجاد تطابق معين للحصول على أفضل تطابق.

تُعطى العناصر <deny> الأولوية على العناصر <allow> .

مثال:

```
<configuration>
...
<system.web>
<authorization>
<allow roles="MyDomainName\SalesDept"
users="MyDomainName\sami,MyMachineName\adel" />
<deny users="*" />
</authorization>
</system.web>
...
</configuration>
```

سوف يسمح الملف التالي للحساب sami من MyDomainName والحساب المحلي adel إضافة إلى جميع المستخدمين من المجموعة SalesDep بالوصول إلى التطبيق ويحجب باقي المستخدمين.

تحديد نمط الوصول في HTTP

يمكن أيضاً باستخدام عناصر <allow> و<deny> التحكم بنمط الوصول باستخدام HTTP باستعمال الواصفة Verb كما في المثال التالي:

```
<configuration>
...
<system.web>
<authorization>
<allow verb="GET" users="*" />
<allow verb="POST" users="MyDomainName\marthasmith" />
<deny verb="POST" users="*" />
</authorization>
</system.web>
...
</configuration>
```

أما في حال الرغبة بإعداد سماحيات الوصول لأكثر من مجلد فرعي أو ملف باستخدام نفس ملف web.config نستخدم العنصر <location> وذلك كما في المثال:

```
<configuration>
...
<system.web> <!-- default for this application -->
<authorization>
<allow verb="GET" users="*" />
<allow verb="POST" users="MyDomainName\marthasmith" />
<deny verb="POST" users="*" />
</authorization>
</system.web>
<location path="mypage.aspx"> <!-- only applies to this file -->
<system.web>
<authorization>
<allow verb="GET" users="*" />
<allow verb="POST" users="MyDomainName\billjones" />
<deny verb="POST" users="*" />
</authorization>
</system.web>
</location>
...
</configuration>
```

يمكن أن نوجه ASP.NET لاستخدام حساب معين بدلاً من الحساب الذي تم التحقق منه من قبل IIS عن طريق العنصر <identity> وذلك كمايلي:

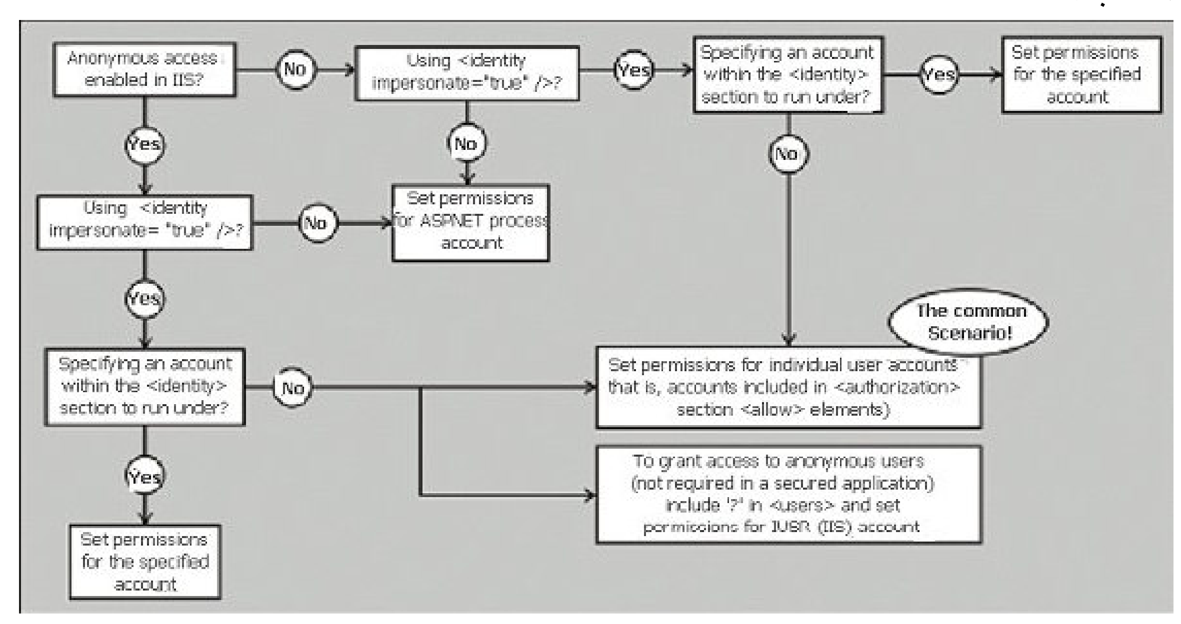
```
<configuration>
...
```

```
<system.web>
<identity impersonate="true"
userName="MyDomainName\MyUserName
password="MyPassword" />
</system.web>
...
</configuration>
```

إعدادات Windows و IIS لحالة التحقق من الهوية باستخدام Windows

ذكرنا أن الوصول باستخدام "Windows" يعتمد على حسابات Windows المتوفرة ل ASP.Net لتتمكن من استخدامها للوصول إلى المصادر.

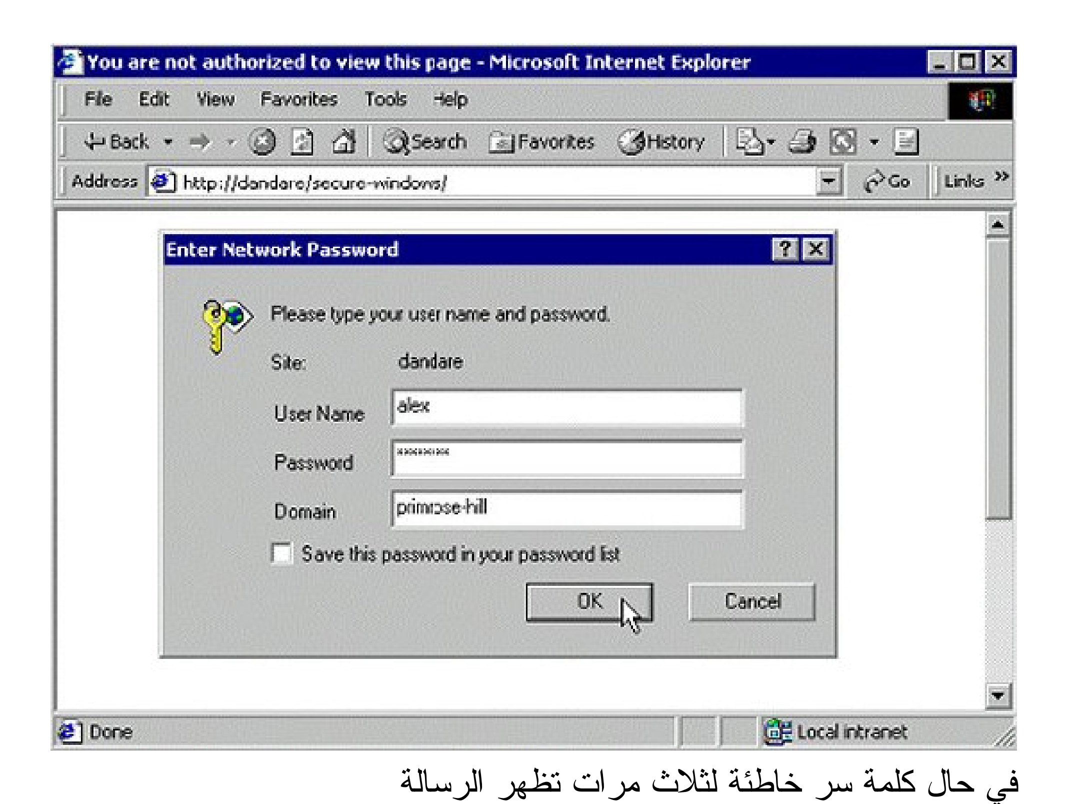
يوضح الشكل التالي أي الحسابات يتم استخدامه للوصول إلى المصادر لتقوم بإعطاء هذا الحساب السماحيات المناسبة:



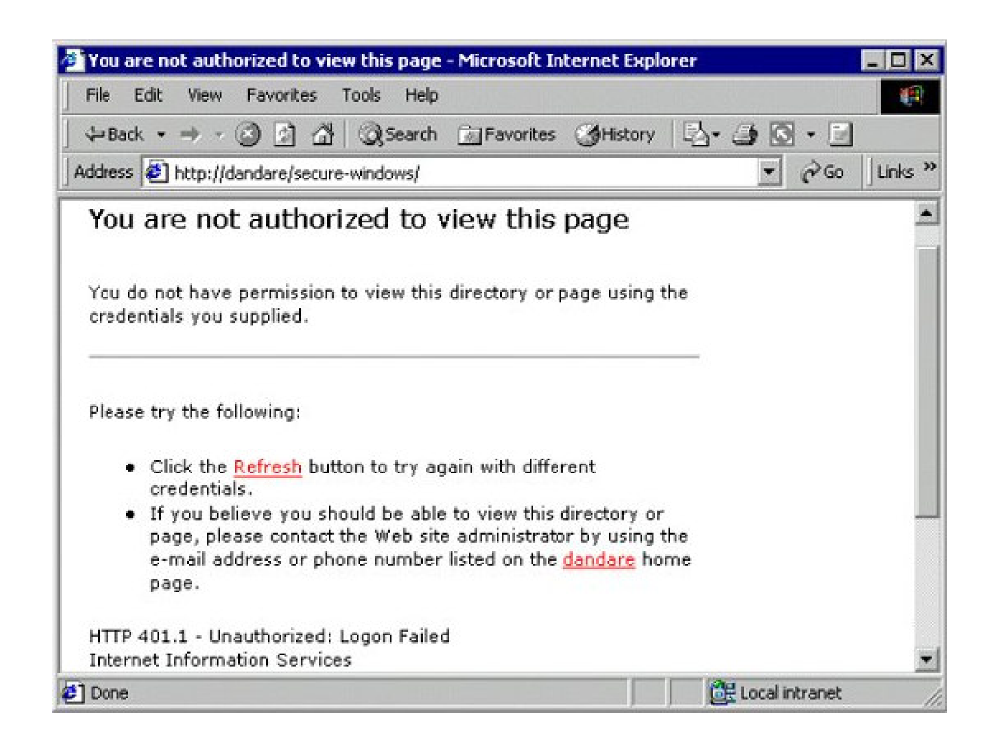
ترجمة الشكل	
Anonymous Access Enabled in IIS?	هل يُسمح بالدخول المجهول الهوية في IIS؟
Using <identity impersonate="True"/>?	تم استخدام<identity impersonate="True"/>
Specifying an account within the <identity> section to run under	تم تحديد حساب ضمن القسم<identity> للعمل باستخدامه

Set permissions for the specified account	تعيين السماحيات للحساب المحدد
Set permissions for ASPNET process account	تعيين السماحيات لحساب الإجراء ASPNET
Set permissions for individual user accounts , that is accounts included in <authorization> section <allow> element	تعيين السماحيات للحسابات المفردة أي الحسابات المضمنة في القسم <authorization> من العنصر <allow>
To grant access to anonymous users (not required in a secured application) include '?' in permission for IUSER (IIS) account	للسماح بالدخول المجهول الهوية (غير مطلوب في تطبيق آمن) يجب تضمين الرمز "؟" في السماحيات الخاصة بحساب IUSER المسمى IIS
The common scenario	الحالة الأكثر شيوعاً
Yes	نعم
No	كلا

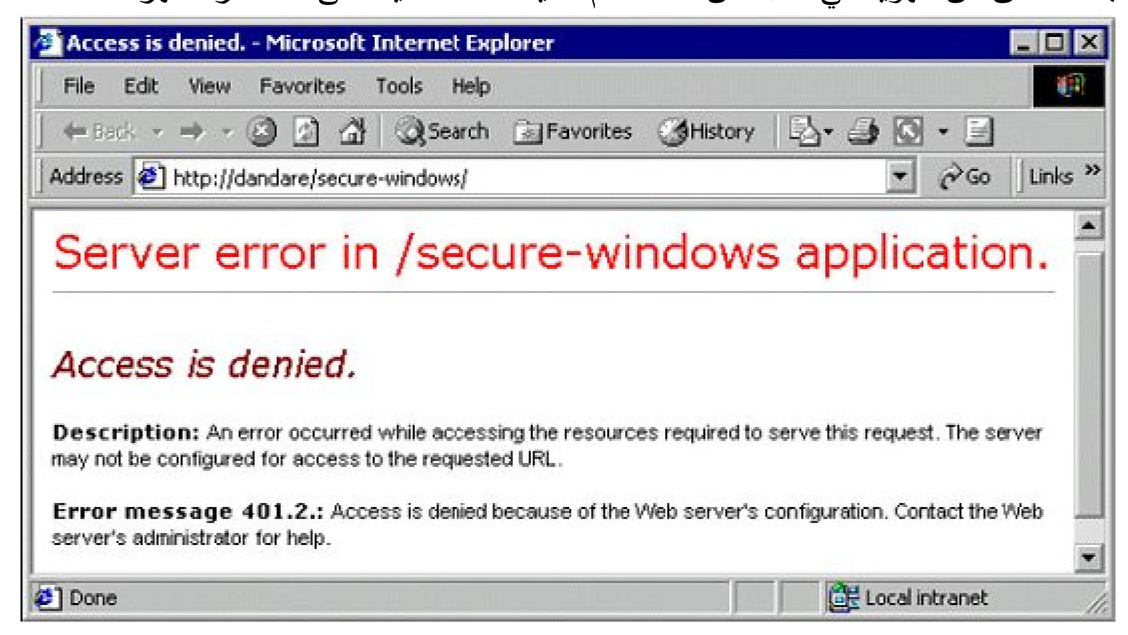
مراحل تسجيل الدخول في تحقق الهوية "Windows":



في حال كلمة سر خاطئة لثلاث مرات تظهر الرسالة



بعد التحقق من الهوية في حال كان المستخدم لا يملك السماحية على المصدر تظهر النافذة



التحقق من الهوية من النمط "Passport"

يقدم استخدام التحقق من الهوية الخاص بـ Windows طريقة آمنة وجيدة للتحكم بالوصول باستخدام ASP.NET، لكن تبرز مشكلته حين نريد استعمال سياسة استخدام توقيع وحيد لأكثر من تطبيق على أكثر من مخدم أو موقع بالأخص إذا كانت متباعدة جغرافياً. يكمن الحل الوحيد في تعريف نفس الحساب على جميع المخدمات أو باستخدام "Forest" مع Active Directory بحيث تكون جميع المخدمات جزء من نفس المؤسسة حتى لو كانت في نطاقات مختلفة.

ولكن هذا الحل يفشل في الحالة التي يكون مطلوباً فيها التحقق من الهوية على أكثر من موقع . كما هي الحال إذا أردنا السماح بالوصول بشكل آلي إلى موقعنا في حال نجح المستخدم بالوصول موقع معروف مثل Hotmail .

يمكن إتمام هذه العملية باستخدام التحقق من الهوية من النمط "Passport" حيث تقدم شركة Microsoft خدمة تسمى "Passport service" يمكن استخدامها للتحقق من هوية المستخدمين على أي موقع يدعم هذه الخاصية. وذلك اعتباراً من أي مكان على الانترنت.

عند تسجيل الدخول على موقع يدعم هذه الخدمة يقوم المستعرض بإرسال المعلومات إلى خدمة Passport التي تقوم بالتحقق من الهوية وتقوم بوضع كعكة على الجهاز .

عندما يحاول المستخدم الوصول إلى موقع آخر يدعم هذه الخدمة يقوم المستعرض بإظهار الكعكة إلى خدمة Passport مثبتاً أنه قد تم التحقق من هويته.

استخدام خدمة Passport غير مجاني ولابد من التسجيل للحصول على هذه الخدمة وتثبيت برنامج خاص على مخدم الوب لتفعيل عمل هذه الخدمة.

بعد عملية التسجيل والإعداد يمكننا ضبط هذا ضمن الملف web.config

```
<configuration>
...
<system.web>
<authentication mode="Passport">
<passport redirectUrl="internal|url" />
</authentication>
</system.web>
...
</configuration>
```

يدعم العنصر <passport> واصفة وحيدة هي redirectURL تحدد المحدد القياسي للصفحة التي سيتم التوجه إليها في حال فشل التحقق من الهوية. تكون هذه القيمة معينة تلقائياً قبل تثبيت خدمة Passport إلى القيمة "Internal".

عند تفعيل خدمة "Passport" يكون إجراء تسجيل الدخول على المخدم كالتالي:

- يقوم المستخدم بطلب مصدر ما من المخدم
- إذا كان هذه المستخدم قد سجل دخوله إلى خدمة Passport سيكون لديه بطاقة مشفرة ضمن كعكة

وسيقوم المخدم بالوصول إلى خدمة Passport للتعرف على هوية المستخدم

- في حال عدم توفر البطاقة المشفرة والمخزنة في كعكة أو في حال انتهاء صلاحيتها يتم إرسال المستخدم إلى صفحة تسجيل الدخول الخاصة بخدمة Passport
- يقوم المستخدم بتسجيل الدخول
- في حال تم التحقق من هويته يتم إعادة تحويله إلى مخدم التطبيق مع البطاقة المعرفة عنه ليتم تخزينها ككعكة على جهاز المستخدم

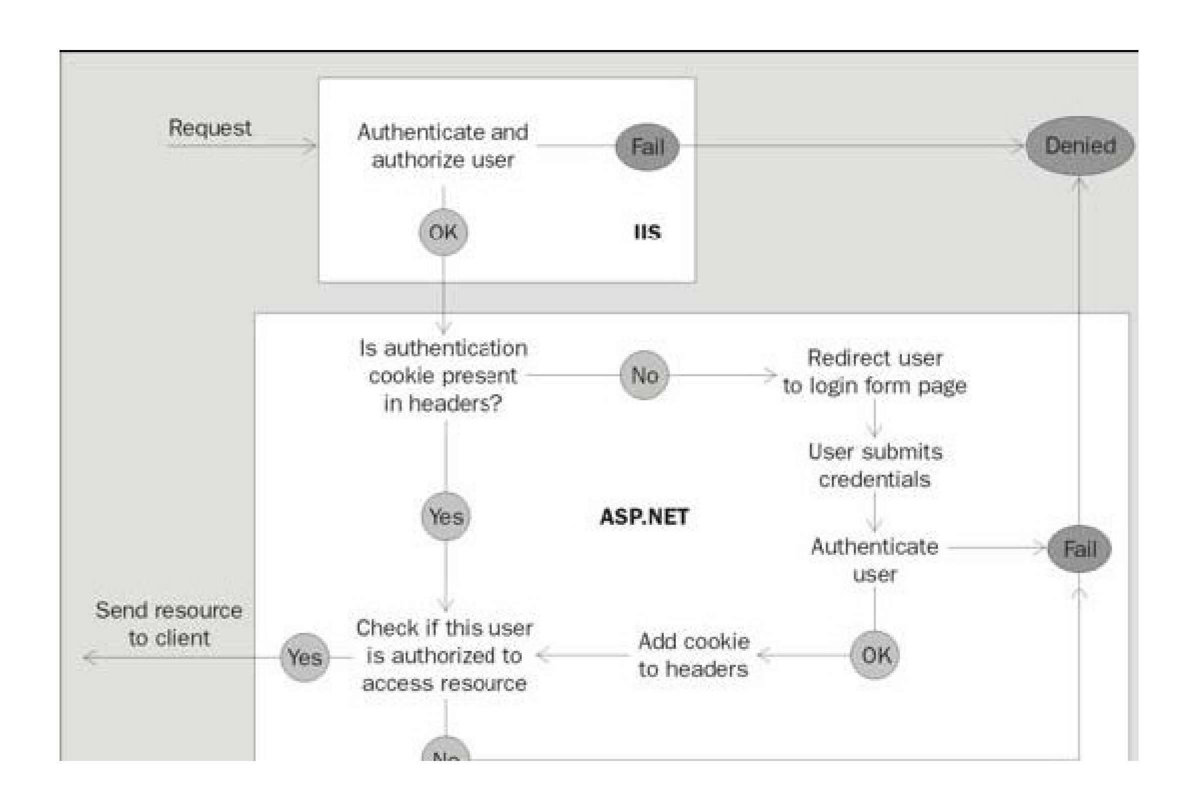
التحقق من الهوية المبني على النماذج

يناسب هذا النوع من التحقق من الحالات التي لا تكون فيها الدرجة المطلوبة من الأمان عالية (أحياناً يشار إلى التحقق من الهوية المبني على النماذج بالتحقق المبني على الكعكات).

الجديد في ASP.NET فيما يتعلق بهذه الطريقة هو أنه تمت أتمتة العديد من العمليات التي كنا نضطر للقيام بها برمجياً عند العمل مع النسخة السابقة (ASP) .

لن نعود مضطرين بهذه الطريقة للمحافظةعلى الشكل الوحيد لشاشة تسجيل الدخول (شاشة تسجيل الدخول الخاصة بWindows) بل يمكننا استبدالها بأي نموذج مخصص جذاب.

يشرح الشكل التالي يشرح إجراء التحقق من الهوية عن طريق النماذج.



ترجمة الشكل	
Request	طلب
Authenticate and authorize user	التحقق من هوية المستخدم ومنحه السماحيات
Is authentication cookie present in header	هل تتضمن ترويسة الطلب الكعكة الخاصة بالتحقق من الهوية
Redirect user to login form page	إعادة توجيه المستخدم إلى صفحة نموذج تسجيل الدخول
User Submit credentials	يرسل المستخدم معلومات التسجيل
Authenticate user	التحقق من هوية المستخدم
Add cookie to header	إضافة الكعكة إلى ترويسة الطلب
Check if user is authorized to access resource	التأكد السماحية للمستخدم بالوصول إلى المورد
Send resource to client	إرسال المورد إلى الزبون
Yes	نعم

No	كلا
Fail	فشل
Denied	ممنوع

بعد أن يتم التحقق من الهوية في IIS يتجه الطلب إلى ASP.NET حيث يتم التأكد من احتواء ترويسة الطلب للكعكة.

يتم توليد الكعكة إذا سبق للمستخدم الوصول إلى التطبيق . أما إذا كانت الكعكة غير متوفرة فمعناه أن هذا المستخدم لم يسجل الدخول إلى التطبيق مسبقاً أو أن صلاحية الكعكة انتهت وبالتالي لابد من توجيه المستخدم إلى نموذج تسجيل دخول مخصص. يدخل المستخدم معلومات اسم الدخول وكلمة السر، ويقوم بإرسال المعلومات إلى التطبيق حيث يتم التحقق منها وإنشاء كعكة وإضافتها إلى ترويسة الطلب وإرسال الطلب إلى المرحلة التالية حيث سيتم التأكد من سماحية وصول المستخدم إلى المصدر المطلوب .

إعداد التحقق من الهوية المبني على النماذج

يتم إعداد التحقق المبني على النماذج عن طريق الملف web.config أيضاً ضمن القسم <authentication> وذلك بالشكل :

```
<configuration>
...
<system.web>
<authentication mode="Forms">
<forms name="cookie-name"
path="cookie-path"
loginurl="url"
protection="All|None|Encryption|Validation"
timeout="number-of-minutes" >
<credentials passwordFormat="Clear|SHA1|MD5">
<user name="user-name" password="user-password" />
<user name="user-name" password="user-password" />
... more users listed here ...
</credentials>
</forms>
</authentication>
<machineKey validationKey="AutoGenerate|key"
decryptionKey="AutoGenerate|key"
validation="SHA1|MD5"/>
</system.web>
...
</configuration>
```

حيث :

- Name يمثل اسم الكعكة
- Path يمثل المسار الذي تكون الكعكة فعالة من أجله (عادة نستخدم "/") للتعبير عن كون الكعكة فعالة لكامل الموقع.
- Login_url تحدد المسار للوصول إلى صفحة تسجيل الدخول.
- Protection تحدد درجة الأمان المطلوبة للكعكة :
  - القيمة All يتم استخدام تشفير مبني على العنصر <machinekey> وتشفير (Triple DES) إذا كان طول المفتاح أكبر من 48 بايت.
  - القيمة None فلا يتم التشفير.
  - القيمة Encryption يتم تشفير الكعكة ولكن لا تتم عملية التحقق من البيانات.
  - Validation تتم عملية التحقق من البيانات ولا تتم عملية تشفير الكعكة.
- Timeout تحدد بالدقائق الوقت اللازم للكعكة لتنتهي صلاحيتها.
- يمكن استخدام العنصر <credentials> لتحديد خوارزمية التشفير المستخدمة لتشفير كلمة السر في ملف web.config يمكن ضمن هذا العنصر إيجاد مجموعة من عناصر <user> التب تحدد المستخدمين القادرين على الوصول إلى المصادر المحمية.
- يمكننا أيضاً تحديد عنصر <machinekey> الذي يحدد المفاتيح وطريقة التشفير التي ستستخدم لتشفير محتوى الكعكة.

القيمة التلقائية لهذا العنصر ستكون من الشكل :

```
<machineKey validationKey="AutoGenerate"
decryptionKey="AutoGenerate"
validation="SHA1" />
```

إعداد التحقق من الهوية المبني على النماذج

مثال على ما سيبدو عليه الملف web.config في حالة تحقق الهوية المبني على النماذج:

```
<configuration>
...
<system.web>
<authentication mode="Forms">
<forms name="MyNewApp" path="/" loginUrl="/main/login.aspx"
protection="All" timeout="30" >
```

```
<credentials passwordFormat="SHA1">
<user name="billjones"
password="87F8ED9157125FFC4DA9E06A7B8011AD80A53FE1" />
<user name="marthasmith"
password="93FB8A49CC350BAEB2661FA5C5C97959BD328C50" />
<user name="joesoap"
password="5469541CA9236F939D889B2B465F9B15A09149E4" />
</credentials>
</forms>
</authentication>
<!-- keys usually only specified for a Web farm -->
<machineKey validationKey="3875f9...645a78ff"
decryptionKey="3875f9...645a78ff"
validation="SHA1" />
</system.web>
...
</configuration>
```

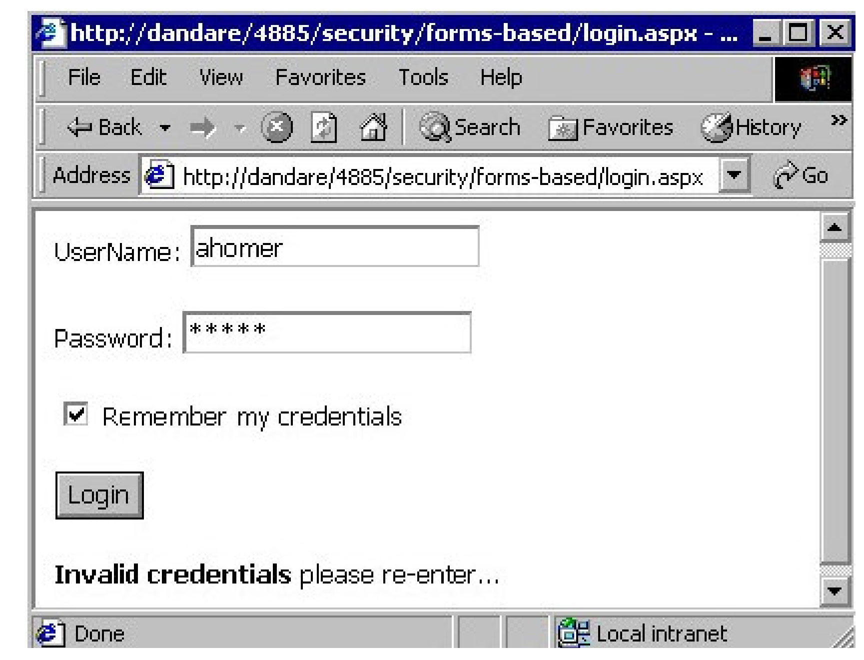
إنشاء نموذج تسجيل الدخول:

بعد إعداد الملف web.config لابد من العمل على إنشاء النموذج الخاص بتسجيل الدخول.

فيما يلي النص البرمجي لمثال بسيط عن مثل هذا النموذج:

```
<%@Page Language="VB" %>
<html>
<body>
<form runat="server">
UserName: <input id="txtUsr" type="text" runat="server" /><p />
Password: <input id="txtPwd" type="password" runat="server" /><p />
<ASP:CheckBox id="chkPersist" runat="server" />
Remember my credentials<p />
<input type="submit" value="Login" runat="server"
onserverclick="DoLogin" />
<div id="outMessage" runat="server" />
</form>
</body>
</html>
```

نتيجة النص البرمجي ستكون كمايلي



إعداد التحقق من الهوية المبني على النماذج

كتابة النص البرمجي لتسجيل الدخول:

بالرغم من كون تحقق الهوية المبني على النماذج تقنية ذكية ولكنها لن تستطيع القيام بكل شيء آلياً إذ لا بد لنا من كتابة نص برمجي لأداء بعض العمليات المطلوبة .

تنتمي جميع الصفوف التي تستخدم في تأمين ASP.NET إلى فضاء الأسماء System.Web.Security . ويدعى الصف الذي يتولى مواضيع التحقق من الهوية باستخدام النماذج FormsAuthentication . يقدم هذا الصف مجموعة من الطرق والخصائص أهمها:

Authenticate	تقوم باختبار اسم المستخدم وكلمة المرور ومطابقتها لتلك المعدة في الملف web.config
redirectFormLoginPage	تقوم بأداء جميع الأعمال المطلوبة (بعد التحقق من الهوية) من إنشاء للكعكة وإضافتها إلى ترويسة الطلب وإرسال الطلب إلى الصفحة التي تمت محاولة الوصول إليها أصلاً .
SignOut	تدمر الكعكة المشفرة بالتالي تقوم بعملية تسجيل خروج المستخدم.
SetAuthCookie	إنشاء الكعكة المشفرة وتضيفها إلى ترويسة الطلب ولكن لا تقوم بتحويل طلب المستخدم .

تقوم بإعادة قيمة الكعكة دون إضافتها إلى ترويسة الاستجابة. هذه الطريقة مفيدة عندما نريد القيام بتخصيص للكعكة قبل تصديرها.	**GetAuthCookie**
تعيد محدد المصدر للصفحة التي طلبها المستخدم قبل توجيهه إلى صفحة تسجيل الدخول.	**GetRedirectUrl**

إذاً يمكننا استخدام هذه الطرق بإضافة النص التالي إلى نصنا البرمجي:

```
Sub DoLogin(objSender As Object, objArgs As EventArgs)
If FormsAuthentication.Authenticate(txtUsr.Value, txtPwd.Value) Then
FormsAuthentication.RedirectFromLoginPage(txtUsr.Value, _
chkPersist.Checked)
Else
outMessage.InnerHtml = "<b>Invalid credentials</b> please re-enter."
End If
End Sub
```

إعداد التحقق من الهوية المبني على النماذج

إلى متى تظل الجلسة فعالة:

تعتبر مدة فعالية الكعكة أحد النقاط الهامة التي يجب الانتباه إليها.

تكون المدة التلقائية لصلاحية الكعكة 30 دقيقة إذا لم يتم تعديلها من خلال ضبط قيمة الواصفة timeout ضمن العنصر <forms> في الملف web.config .

يتم إنهاء صلاحية الكعكة أيضاً عند إغلاق تطبيق المستعرض. بالطبع سوف يتم تحديث الكعكة مع كل استجابة من المخدم لإطالة حياتها . لذلك يحسب الزمن من آخر وصول إلى الصفحة .

تقدم ASP.NET خيار الإبقاء على الكعكة بين الجلسات. فعند استدعاء الطريقة RedirectFromLoginPage نقوم بتحديد قيمة منطقية للمعامل الثاني الذي تأخذه هذه الطريقة وهو في مثالنا السابق قيمة الخاصة Checked لعنصر التحكم checkbox على صفحة تسجيل الدخول:

```
FormsAuthentication.RedirectFromLoginPage(username, persist-cookie)
```

عند تمرير القيمة True يتم إنشاء كعكة تملك زمن صلاحية طويل جداً (50 عاماً) مما يسمح للمستخدم بعدم الاضطرار إلى تسجيل الدخول مجدداً.

تقوم أغلب المستعرضات الحديثة بتخزين الكعكات على أساس المستخدم مما يشكل مشكلة أمنية لأنه من السهل اختطاف كعكة وبالتالي سوف يتمتع الخاطف بالوصول إلى الموقع خلال فترة حياة الكعكة. لذلك يفضل عدم تطبيق هذا السيناريو إلى في الحالات التي لا تتطلب أمن عالي.

على أي حال نستطيع ضمان تدمير الكعكة قسرياً باستخدام الطريقة SignOut:

```
FormsAuthentication.SignOut()
```

لكن هذه الطريقة غير قادرة على اكتشاف كعكة مسروقة وتدميرها، لذلك يجب الابتعاد عن استخدام الكعكات ذات العمر الطويل .

إعداد السماحيات للأمان المبني على النماذج

تكلمنا حتى الآن عن التحقق من الهوية المبني على النماذج لكننا لم نحدد ما هي المصادر التي يسمح للمستخدمين بالوصول إليها. يمكن إضافة القسم <authorization> إلى الملف web.config وإلا سيتم تطبيق السماحيات التلقائية المحددة في الملف machine.config والذي يحتوي فيه القسم <authorization> ما يلي:

```
<authorization>
<allow users="*" />
</authorization>
```

لذلك وللسماح فقط لمجموعة معينة من المستخدمين بالدخول نكتب في الملف :web.config

```
<configuration>
...
<system.web>
<authorization>
<allow users="billjones,marthasmith,joesoap" verb="GET" />
<allow users="marthasmith" verb="POST" />
<deny users="?" />
</authorization>
</system.web>
...
</configuration>
<authorization>
<deny users="?" />
</authorization>
```

يجب ألا نتحمس لتطبيق سماحيات ACL الخاصة بـــ Windows على الموارد، فحتى لو كان هؤلاء المستخدمين معرفين في Windows وتم ضبط سماحيات لهم فلن يتم استخدام أي من هذه الحسابات عند استخدام التحقق من الهوية المبني على النماذج.

يصبح من الصعب مع عدد كبير من المستخدمين، التعديل يدوياً على المستخدمين ضمن القسم <credentials> من الملف web.config، لذا نحتاج مثلاً لتخزين هذه المعلومات في مكان آخر أو في ملف XML أو قاعدة بيانات أو حتى في Active Directory، ولا بد عندها من تطوير آلية المقارنة بين المعلومات المدخلة في نموذج تسجيل الدخول وما هو موجود ضمن قاعدة البيانات أو في ملف XML وذلك باستخدام التقنيات التي تعرفنا عليها في الجزء الخاص بالاتصال بمصادر البيانات المختلفة.

خاتمة

لاحظنا من خلال ما اطلعنا عليه حتى الآن تركيزنا على التحكم بالوصول إلى المصادر بالاعتماد على مبدأ التعرف بصورة فردية على المستخدم باستخدام تحقق الهوية ثم التأكد من سماحيات المستخدم للمصادر المحددة .

لكن هناك بعض الحالات التي نحتاج فيها إلى تحديد اسم المستخدم المصرح له بالدخول من خلال النص البرمجي. يتم أداء تلك العمليات باستخدام مجموعة من الأغراض منها User , IPrincipal. فيما يلي بعض العمليات التي نستطيع القيام بها.

لاستعادة اسم المستخدم يمكننا استخدام الكائن User

```
strUserName = User.Identity.Name
```

لتحديد كون تحقق الهوية قد تم بنجاح نستخدم الخاصة IsAuthenticated

```
blnAuthenticated = User.Identity.IsAuthenticated
```

و لتحديد نوع تحقق الهوية المعتمد نستخدم

```
strAuthType = User.Identity.AuthenticationType
```

و في حال كان نوع التحقق من الهوية المستخدم هو Windows يمكننا تحديد فيما إذا كان المستخدم ينتمي إلى مجموعة محددة باستخدام الطريقة IsInRole

```
blnResult = User.IsInRole("MyDomainName\SalesDept")
```

الفصل السابع عشر

عنوان الموضوع:

تطبيقات الوب في asp.net و ملفات التهيئة.

الكلمات المفتاحية:

أنظر الـــ Glossary المرفق

ملخص:

طورت ASP.NET مفهوم تطبيقات الوب و ضمنته الكثير من المزايا كما قدمت آليات أكثر مرونة في ضبط إعدادات ملفات التهيئة.

ضمن هذه الجلسة سنقوم بالتعرف على مفهوم تطبيقات الوب و كيفية إنشاءها وتنسيق ملفات التهيئة وكيفية الاستفادة منها

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- كيفية إنشاء مجلدات التطبيقات و تسجيل المكونات
- بنية التطبيق و الملفات الأساسية
- تنسيق الملف GLOBAL.ASAX
- تنسيق ملفات التهيئة وكيفية الاستفادة منها

تطبيقات الوب في ASP.NET و الملف glogal.asax

دعمت نسخة asp مبدأ تطبيقات الوب حيث أنها كانت على شكل مجموعة من ملفات asp. إضافة إلى ملف باسم global.asa .

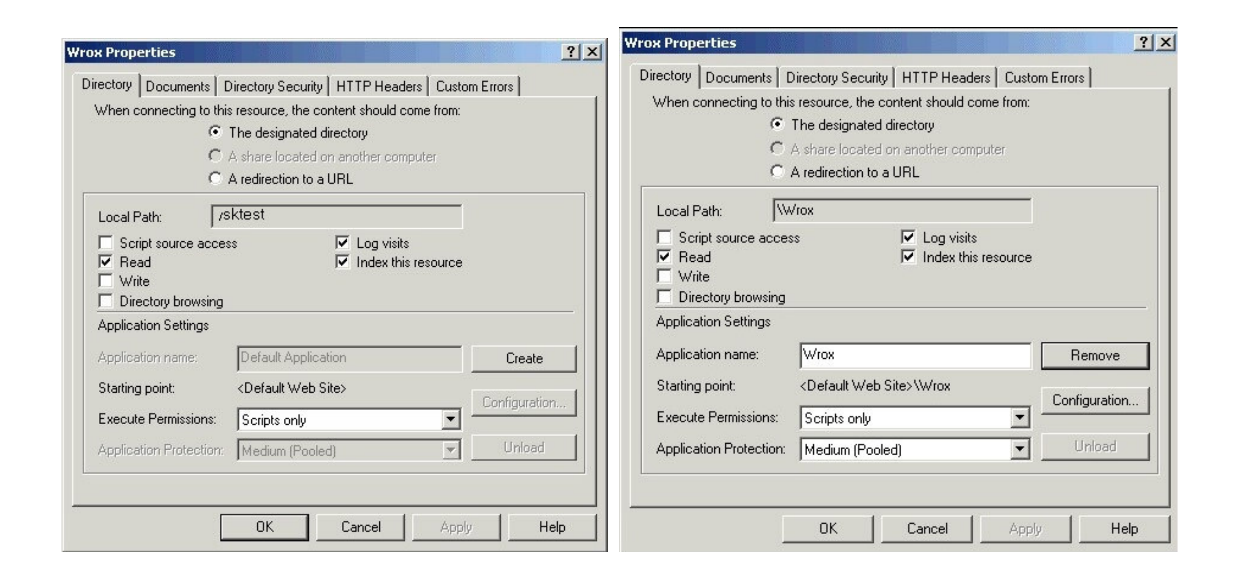
قامت Asp.net بتوسيع هذا المفهوم و أضافت مصادر أخرى كصفحات ASP.NET و خدمات الوب ، عناصر التحكم الخاصة بالمستخدم ، وملف إعداد البيانات و الملف global.asax و ملفات أخرى عديدة تم إنشاؤها إما من قبل المستخدم أو مضمنة في ASP.Net.

ضمن هذه الجلسة سنقوم بالتعرف على مفهوم تطبيقات الوب و كيفية إنشاءها و تنسيق الملف global.asax و الملفات web.config، machine.config و كيفية الاستفادة منها.

عرفنا في جلسات سابقة المجلدات الافتراضية و ذكرنا فوائدها ورأينا كيف بإمكاننا إنشاء مجلد افتراضي في IIS .

الآن و بفرض أن لدينا مجلد افتراضي كيف سنقوم بتحويل هذا المجلد إلى تطبيق وب ؟

يمكننا ببساطة أن ننقر بالزر الأيمن على المجلد الافتراضي نختار الخيار Properties عندها ستظهر شاشة كما في الشكل إلى اليمين:



في تلك الشاشة يمكننا رؤية إعدادات المجلد ، قيمة الخاصة Local Path التي تحدد المسار المحلي ، و السماحيات المطبقة على هذا المجلد .

نلاحظ في حالة شكلنا أعلاه أن السماحياتالمتوفرة هي القراءة و تسجيل الزيارات فقط إضافة إلى تفعيل الفهرسة على هذا المورد.

و في الجزء الأسفل ، في القسم المعرف كـــ Application setting فنلاحظ أنه يمكننا تفعيل صفة تطبيق الوب بالنقر على الزر Create عندها ستتغير بعض العناصر على النافذة لتصبح كما في الشكل الظاهر على اليسار.بعدها يكفي نقر الزر Apply لتثبيت التغييرات.

أما في حال الرغبة بإزالة صفة تطبيق الوب عن المجلد الافتراضي فيكفي نقر الزر Remove.

تطبيقات الوب في ASP.NET

تستخدم ASP.NET تطبيقات الوب الخاصة ب IIS لتعريف نطاقات خاصة للطبيقات . كل نطاق لتطبيق مفصول عن النطاقات الأخرى ، آمن و لا يتشارك بالذاكرة مع النطاقات الأخرى. أي لن يتم تشارك بيانات مثل Session و Application (و هي بيانات سنتكلم عنها باستفاضة لاحقاً) مع بقية النطاقات .
عملية الفصل هذه ستجعل من أي خلل في عمل أحد التطبيقات في نطاق ما غير مؤثر و منفصل عن بقية التطبيقات.
تتألف تطبيقات وب ASP.NET عادة من ثلاثة أنماط من المصادر التي يمكن للمستخدم إنشاؤها إضافة إلى تلك القياسية المبيتة في ASP.NET و الصفحات و خدمات الوب:

BIN هذا المجلد يتوضع مباشرة ضمن المجلد الجذر و يستخدم لاحتواء معلومات المجمعة التي تستخدمها التطبيقات.

Global.asax هذا الملف هو بديل للملف global.asa الذي كان مستخدماً مع ASP و هو يسمح بتنفيذ نص برمجي للأحداث في ASP.NET و إسناد قيم إلى المتغيرات وذلك على مستوى تطبيق الوب كاملاً.

Web.config: يمكّن إعطاء إعدادات خاصة لكل تطبيق اعتماداً على مستوى الأمان المطلوب. يمكن للإعددات المثبتة في هذا الملف التغطية على الإعددات العامة المعينة في الملف machine.config الذي يحدد الإعددات العامة التي تنطبق على جميع التطبيقات في بيئة ASP.NET المثبتة على المخدم.

تسجيل المكونات :

عوضاً عن الاعتماد على سجلات النظام تعتمد ASP.NET مجلد خاص هو Bin لتسجيل المكونات كجزء من التطبيق.
كانت هذه العملية تتم سابقاً (في ASP) بتسجيل المكونات عن طريق النظام على سبيل المثال كنا نضطر لاستخدام الأداة REGSVR32.EXE لتسجيل المكونات أو عن طريق +COM من الخيار Component Service في خيار الأدوات الإدارية ضمن لوحة التحكم.

```
<%
Dim myObject
Set myObject = Server.CreateObject("Example.DataAccess")
%>
```

تطبيقات الوب في ASP.NET- تسجيل المكونات

طرح موضوع استخدام تسجيل المكونات عن طريق النظام مجموعة من التساؤلات حول قابلية النقل بين مخدمات متعددة و استبدال المكونات أثناء العمل إضافة إلى موضوع الاضطرار للحصول على سماحية الوصول محلياً إلى المخدم لتسجيل المكونات مع عدم إمكانية تشغيل أكثر من نسخة مختلفة من المكون لأكثر من تطبيق.

- تم حل كل تلك المشاكل السابقة في ASP.NET و أصبح بالإمكان اتمام عمليات التحديث و النقل و استخدام أكثر من نسخة من المكون ضمن هذه البيئة.
- لا تتطلب NET. وصول محلي إلى المخدم لتسجيل المكونات. بكل بساطة يكفي نسخ النص المكتوب بلغة التجميع إلى المجلد Bin حيث يتم شحنها من قبل ASP.NET و تصبح متوفرة للاستخدام من قبل تطبيقاتنا.
- لا يوجد سوى مجلد واحد باسم Bin لكل تطبيق و بالتالي لن تون مكونات المسجلة في تطبيق ما إلا لهذا التطبيق .
- لا نحتاج لاستبدال أي مكون في ASP.NET إلى إيقاف IIS (بعكس ASP) بكل بساطة يمكن التعديل على الملف التجميعي المرتبط بالمكون أو التعديل على التطبيق .
- تمنح ASP.NET هذه الخاصية لأنها تقوم بالتنصت على أية تعديلات على المجلد Bin و ما أن تكتشف أي تعديل تقوم فوراً بإنشاء نطاق تطبيق جديد و تبدأ بتلقي الطلبات عليه أما بالنسبة إلى نطاق التطبيق قبل التعديل فسيتم الانتظار لحين إنهاءه التعامل مع الطلبات الحالية و إزالته بعد ذلك.

تنسيق الملف global.asax

يتبع الملف global.asax تنسيق مشابه لصفحات ASP.NET و يأخذ عادة الصيغة العامة التالية:

```
<%@ [Directive] [attribute]=[value] %>
<Script runat="server">
[Application Event Prototypes]
</Script>
```

الموجهات:

يدعم global.asax تعليمات صفحات ASP.NET إضافة إلى تعليمات خاصة تستخدم في ترجمة هذا الملف . من أهم التوجيهات التي يدعمها global.asax التالية:

- Application و التي تسمح بتحديد الصف القاعدي الذي سيقوم global.asax باستخدامه، إضافة إلى أنه يدعم خيار توثيق بسيط.

هذه الميزات مدعومة من خلال واصفتين هما inherits و Description.

- تحدد الواصفة Inherits اسم الصف الذي سيتم استخدامه من قبل global.asax كصف قاعدي لجميع المثائل المترجمة من هذا الملف.تفيد هذه الميزة في إضافة طرق أو خصائص إلى global.asax .
- تحدد الواصفة Description وصف بسيط للملف global.asax

```
<%@ Application Description="A sample global.asax Description” %>
```

- Import يمكن هذا الموجه من استيراد فضاء أسماء لاستخدامه ضمن global.asax و يقوم بإعطائنا تأهيل كامل لصفوف هذا الفضاء ضمن global.asax.

لاشتخدام الموجه Import يجب ضمان كون الملف التجميعي الحاوي لفضاء الأسماء متوفر. يمكن إضافة الملف التجميعي في حال عدم توفره باستخدام الموجه Assembly .

- Assembly يستخدم هذا الموجه لتحديد الملف التجميعي الحاوي على الصفوف التي نود استخدامها في تطبيق ASP.NET . الواصفة الأساسية للموجه Assembly هي Name حيث يتم بواسطتها تحديد اسم الملف التجميعي.

يختلف الموجهان Import و Assembly إذ يعتبر الأول أن الملف التجميعي الحاوي على الصفوف المطلوبة متوفر للتطبيق . بينما يخبر الثاني Asp.NET أن هناك ملف تجميعي يجب شحنه.

تنسيق الملف global.asax

التصريح عن النص البرمجي:

- يتم التصريح عن النص البرمجي في global.asax كالعادة باستخدام التأشيرة

```
<Script runat=’server’>
```

- كما يمكن استخدام SSI (النصوص المدرجة من قبل المخدم) باستخدام التعبير #include بالشكل:

```
<!--#Include [File | Virtual]="Path to file" -->
```

حيث تمثل file المسار إلى الملف اعتماداً على نظام الملفات العادي و virtual المسار إلى الملف عبر مخدم الوب.

ستتم إضافة محتوى الملف قبل ترجمته إلى ملف global.asax.

يمكن استخدام هذه الميزة في تحديد اسماء ملفات تحوي معلومات نرغب بمشاركتها ضمن أكثر من ملف global.asax في أكثر من تطبيق.

- كذلك يمكننا استخدام التأشيرة <object> للتصريح عن أغراض بمجال عمل Session و

Application . يمكن استخدام هذه التأشيرة لإنشاء ملفات NET. تجميعية و أو أغراض COM محددة ب ProgID أو CLSID.

أنواع الأغراض التي يمكن إنشاؤها يمكن تعريفها باستخدام ثلاث واصفات هي:

Class، progid و classid حيث يمكن استخدام أحد هذه الواصفات فقط ضمن التأشيرة <object>.

```
<object id="appData" runat="server"
class="System.Data.DataSet" scope="Application"/>
```

نرى في المثال التالي أننا قمنا بتعريف متحول بالاسم ,"appData" و الذي يشكل مثيلاً من الصف System.Data.DataSet.

تحدد الواصفة Scope في مجال سوف يعمل هذا المتحول و بالتالي تحدد حياة المتحول هل ستكون مرتبطة بالتطبيق في حال اخترنا القيمة "Application" أو بالجلسة الحالية في حال اخترنا "Session" .

التصريح عن النص البرمجي:

- يتم التصريح عن النص البرمجي في global.asax كالعادة باستخدام التأشيرة <Script runat='server'>

- كما يمكن استخدام SSI (النصوص المدرجة من قبل المخدم) باستخدام التعبير #include

كذلك يمكننا استخدام التأشيرة <object> للتصريح عن أغراض بمجال عمل Session و Application

ملفات التهيئة في ASP.NET

ملفات التهيئة في ASP.NET هي ملفات نصية مبنية بتنسيق XML .

تعطى هذه الملفات الاسم web.config . يمكن لهذه الملفات أن تظهر في أي مجلد على مخدم الوب .

يطبق كل ملف web.config إعداداته على المجلد الذي يحويه و كل المجلدات الأبناء ضمن هذا المجلد.

يمكن للإعدادات في المجلدات الأبناء أن تعدل بعض الإعدادات للمجلدات الأباء لها.

يحدد ملف التهيئة الجذر الإعدادات التي ستطبق على جميع التطبيقات ما لم يتم تجاوزها باستخدام ملفات التهيئة للمجلد نفسه.

تقوم ASP.NET بإعداد IIS بحيث يمنع الوصول المباشر لملفات web.config للتأكد من أن أحداً لن يستطيع الوصول إليها لتعديلها إذا كان لا يملك السماحية المناسبة.

على سبيل المثال يمكن أن تكون الإعدادات الخاصة بمحدد المصدر القياسي

http://myServer/myApplication/mydir/page.aspx مطبقة باستخدام ملفات web.config بالترتيب التالي:

الإعدادات لملف التهيئة الأساسي يطلق على الملف الاسم machine.config

C:\WinNT\Microsoft.NET\Framework\v.1.00\config\machine.config

بالمعلومات الموجودة في الملف:wwwroot سيتم تجاوز هذا الملف بالنسبة للموقع الجذر

C:\inetpub\wwwroot\web.config

و التي يمكن تجاوزها بالنسبة لتطبيق معين باستخدام الملف:

D:\MyApplication\web.config

و التي يمكن أن يتم تجاوزها بالنسبة لمجلد ما ضمن تطبيق باستخدام الملف

D:\MyApplication\MyDir\web.config

إذا وجد الملف web.config في المجلد الجذر للموقع سيتم تطبيق إعداداته على جميع التطبيقات ضمن هذا الموقع.

ملفات التهيئة في ASP.NET هي ملفات نصية مبنية بتنسيق XML .

تعطى هذه الملفات الاسم web.config . يمكن لهه الملفات أن تظهر في أي مجلد على مخدم الوب .

يطبق كل ملف web.config إعداداته على المجلد الذي يحويه و كل المجلدات الأبناء ضمن هذا المجلد.

يمكن للإعدادات في المجلدات الأبناء أن تعدل بعض الإعدادا ت للمجلدات الأباء لها.

يحدد ملف التهيئة الجذر الإعدادات التي ستطبق على جميع التطبيقات ما لم يتم تجاوزها باستخدام ملفات التهيئة للمجلد نفسه.

تقوم ASP.NET بإعداد IIS بحيث يمنع الوصول المباشر لملفات web.config للتأكد من أن أحداً لـــن يستطيع الوصول إليها لتعديلها إذا كان لا يملك السماحية المناسبة.

أقسام التهيئة و معالجات أقسام التهيئة

كما ذكرنا فإن ملف web.config هو ملف بتنسيق XML يمكنه احتواء مكونات ملفات XML القياسية بما فيها العناصر و التأشيرات ، النصوص ، و البيانات .

يمكن أن يكون ترميز الملف ANSIأو Utf-8 أو Unicode حيث يكتشف النظام ذلك تلقائياً.

العنصر الجذر لملف Web.config هو التأشيرة <configuration> حيث يأخذ محتوى هذا الملف الشكل

```
<configuration>
<!- Configuration settings would go here. -->
</configuration>
```

تحتوي التأشيرة <configuration> على ثلاث أنواع من العناصر.

- التصريحات عن معالجات أقسام التهيئة
- مجموعات أقسام التهيئة
- إعدادات أقسام التهيئة

معالجات أقسام التهيئة:

لا تضع ASP.NET أي افتراضات ضمن ملف web.config فيما يتعلق بتنسيق الملف أو الإعدادات المدعومة ضمنه. عوضاً عن ذلك تقوم بتوكيل معالجة بيانات الملف web.config إلى معالجات أقسام التهيئة ، و هي صفوف ضمن إطار عمل NET. التي تستخدم الواجهة البينية IConfigurationSectionHandler .

لا بد من التصريح عن هذا الصف على الأقل ضمن الملف الأساسي machine.config حيث يتم وراثة دعم هذا الصف في جميع المجلدات الفرعية.

يتم التصريح عن هذه المعالجات ضمن ملف web.config باستخدام التأشيرة <configSection> يمكن أن يتم تنسيق الأقسام ضمن مجموعات منطقية باستخدام مجموعات الأقسام .

كل قسم في الملف web.config يحتوي معلومات التهيئة إضافة إلى مثيل عن الصف IConfigurationSectionHandler ليقوم بمعالجته.

أقسام التهيئة و معالجات أقسام التهيئة

مجموعات أقسام التهيئة:

توفر ASP.NET إمكانية التجميع ببنية هرمية للأقسام لأغراض تنظيمية فقط . يمكن أن تظهر التأشيرة <SectionGroup> ضمن التأشيرة <configSections> أو ضمن تأشيرة <sectionGroup>أخرى.

على سبيل المثال تظهر جميع معالجات الأقسام ضمن مجموعة الأقسام <system.web> .

أقسام التهيئة :

توضع الإعدادات الخاصة بالتهيئة ضمن تأشيرات الأقسام .

لا بد من وجود معالج قسم معرف لكل قسم . على سبيل المثال التأشيرة <httpModules> تحدد الإعدادات الخاصة بالوحدات النمطية لHTTP . و يتولى الصف System.Configuration.HttpModulesConfigurationHandler عملية تفسير المحتوى ضمن التأشيرة <httpModules> .

يجب الملاحظة هنا أنه يجب أن يكون لمعالج القسم و الفسم نفس مجموعة القسم.

كذلك لا بد من الانتباه بأن الأسماء للتأشيرات حساسة لنمط الأحرف (كبيرة صغيرة).

مثال:

```
<configuration>

 <configSections>
  <sectionGroup name="system.web">
   <section
    name="httpModules"
    type="System.Web.Configuration.HttpModulesConfigurationHandler,System.Web"
   />
  </sectionGroup>
 </configSections>

 <system.web>
  <httpModules>
   <add
    name="CookielessSession"
    type="System.Web.SessionState.CookielessSessionModule,System.Web"
   />
   <add
    name="OutputCache"
    type="System.Web.Caching.OutputCacheModule,System.Web"
   />
   <add
    name="Session"
    type="System.Web.SessionState.SessionStateModule,System.Web"
   />
   <add
    name="WindowsAuthentication"
    type="System.Web.Security.WindowsAuthenticationModule,System.Web"
   />
   <add
    name="FormsAuthentication"
    type="System.Web.Security.FormsAuthenticationModule,System.Web"
   />
   <add
```

```
      name="PassportAuthentication"
      type="System.Web.Security.PassportAuthenticationModule,System.Web"
    />
    <add
      name="UrlAuthorization"
      type="System.Web.Security.UrlAuthorizationModule,System.Web"
    />
    <add
      name="FileAuthorization"
      type="System.Web.Security.FileAuthorizationModule,System.Web"
    />
  </httpModules>
</system.web>

</configuration>
```

مجموعات أقسام التهيئة:

توفر ASP.NET إمكانية التجميع ببنية هرمية للأقسام لأغراض تنظيمية فقط . يمكن أن تظهر التأشيرة <SectionGroup> ضمن التأشيرة <configSections> أو ضمن تأشيرة <sectionGroup>أخرى.
على سبيل المثال تظهر جميع معالجات الأقسام ضمن مجموعة الأقسام <system.web> .

أقسام التهيئة :

توضع الإعدادات الخاصة بالتهيئة ضمن تأشيرات الأقسام .
لا بد من وجود معالج قسم معرف لكل قسم . على سبيل المثال التأشيرة <httpModules> تحدد الإعدادات الخاصة بالوحدات النمطية لـHTTP . و يتولى الصف System.Configuration.HttpModulesConfigurationHandler عملية تفسير المحتوى ضمن التأشيرة <httpModules> .
يجب الملاحظة هنا أنه يجب أن يكون لمعالج القسم و الفسم نفس مجموعة القسم.
كذلك لا بد من الانتباه بأن الأسماء للتأشيرات حساسة لنمط الأحرف .

استخدام الموقع و المسار

كما ذكرنا يتم بشكل تلقائي توريث جميع الإعدادات الخاصة بالتهيئة إلى المجلد الذي يوجد فيه و جميع المجلدات الفرعية ضمنه.
يمكن تطبيق إعدادات خاصة لمسار فرعي ما باستخدام ملف التهيئة الحالي باستخدام التأشيرة <location> مع واصفة path مناسبة.
في حال كان ملف التهيئة الحالي هو machine.config يمكننا تطبيق إعدادات خاصة بالمجلدات

الافتراضية أو التطبيقات. اما في ملفات web.config العادية فيمكنك ضبط الإعدادات لملف معين ، مجلد ، مجلد افتراضي أو تطبيق.

مثال :

```
<configuration>

  <location path="EnglishPages">
    <system.web>
      <globalization
        requestEncoding="iso-8859-1"
        responseEncoding="iso-8859-1"
      />
    </system.web>
  </location>

  <location path="EnglishPages/OneJapanesePage.aspx">
    <system.web>
      <globalization
        requestEncoding="Shift-JIS"
        responseEncoding="Shift-JIS"
      />
    </system.web>
  </location>

</configuration>
```

إقفال إعدادات التهيئة :

بالإضافة إلى تحديد إعدادات خاصة بواسطة التأشيرة <location> يمكننا تحديد إعدادات أمان معينة بحيث لا يتم تغيير هذه إعدادات التهيئة المحددة في هذا الملف بواسطة إعدادات تهيئة من ملف آخر .

لإقفال مجموعة إعدادات يمكننا استخدام الواصفة allowOverride على مواقع محددة و تعيينها إلى false.

في المثال التالي يقوم نص التهيئة بإقفال إعدادات التمثيل لتطبيقين مختلفين.

```
<configuration>

  <location path="app1" allowOverride="false">
    <system.web>
      <identity impersonate="false" userName="app1" password="app1pw" />
    </system.web>
  </location>

  <location path="app2" allowOverride="false">
    <system.web>
      <identity impersonate="false" userName="app2" password="app2pw" />
    </system.web>
  </location>
```

```
</configuration>
```

عند محاولة تجاوز هذا الإعداد و تغييره باستخدام ملف تهيئة مختلف من الشكل :

```
<configuration>

  <system.web>
    <identity userName="developer" password="loginpw" />
  </system.web>

</configuration>
```

سوف يعيد النظام رسالة خطأ.

كما ذكرنا يتم بشكل تلقائي توريث جميع الإعدادات الخاصة بالتهيئة إلى المجلد الذي يوجد فيه و جميع المجلدات الفرعية ضمنه.

يمكن تطبيق إعدادات خاصة لمسار فرعي ما باستخدام ملف التهيئة الحالي باستخدام التأشيرة <location> مع واصفة path مناسبة.

في حال كان ملف التهيئة الحالي هو machine.config يمكننا تطبيق إعدادات خاصة بالمجلدات الافتراضية أو التطبيقات. اما في ملفات web.config العادية فيمكنك ضبط الإعدادات لملف معين ، مجلد ، مجلد افتراضي أو تطبيق.

إقفال إعدادات التهيئة :

بالإضافة إلى تحديد إعدادات خاصة بواسطة التأشيرة <location> يمكننا تحديد إعدادات أمان معينة بحيث لا يتم تغيير هذه إعدادات التهيئة المحددة في هذا الملف بواسطة إعدادات تهيئة من ملف آخر .

لإقفال مجموعة إعدادات يمكننا استخدام الواصفة allowOverride على مواقع محددة و تعيينها إلى false.

الأقسام القياسية في ملف التهيئة لـــ ASP.NET

هناك مجموعة من معالجات الأقسام القياسية التي تستخدم لمعالجة إعدادات التهيئة ضمن ملفات web.config . الجدول التالي يبين الأقسام القياسية و ووصف لكل منها.

اسم القسم	الوصف
<httpModules>	هذا القسم مسؤول عن ضبط الإعدادات الخاصة بالوحدات النمطية لـــhttp ضمن التطبيق. حيث أن هذه الوحدات تشترك في عملية معالجة جميع

	الطلبات الموجهة إلى التطبيق. من الاستخدامات الشائعة نواحي الأمان و عمليات تسجيل الدخول.
<httpHandlers>	هذا القسم مسؤول عن المحددات الواردة إلى الصفوف IHttpHandler . لا ترث المجلدات الفرعية هذه الإعدادات ، كما أن هذا القسم مسؤول أيضاً عن توجيه المحددات URL القادمة إلى صفوف IHttpHandlerFactory.
<sessionState>	هذا القسم مسؤول عن إعدادات الوحدة النمطية الخاصة بحالة الجلسة
<globalization>	هذا القسم مسؤول عن الإعدادات الخاصة بالعالمية و التخصيص المحلي.
<compilation>	هذا القسم مسؤول عن إعدادات الترجمة المستخدمة في ASP.NET.
<trace>	هذا القسم مسؤول عن إعدادات خدمة التتبع في ASP.NET
<processModel>	هذا القسم مسؤول عن إعدادات نموذج الإجرائية ASP.NET في نظام مخدم وب IIS
<browserCaps>	هذا القسم مسؤول عن التحكم بإعدادات مكون قدرات المستعرض

استرجاع معلومات التهيئة

تسمح Asp.NET للمطورين بالوصول إلى إعدادات التهيئة من ضمن التطبيق بالوصول مباشرة إلى الإعداد أو باستخدام الوصلات البينية البرمجية التطبيقية (API) .

المثال التالي يبين صفحة تقوم بالوصول إلى قسم التهيئة <browserCap> باستخدام الخاصة Browser للصف System.Web.HttpRequest .

```
<%@ Page Language="VB" %>

<html>
<body style="font: 10pt verdana">

  <h3>Retrieving Browser Capabilities</h3>

  Boolean ActiveXControls = <%=Request.Browser.ActiveXControls.ToString()%><br>
  Boolean AOL = <%=Request.Browser.AOL.ToString()%><br>
  Boolean BackgroundSounds = <%=Request.Browser.BackgroundSounds.ToString()%><br>
  Boolean Beta = <%=Request.Browser.Beta.ToString()%><br>
  String Browser = <%=Request.Browser.Browser%><br>
  Boolean CDF = <%=Request.Browser.CDF.ToString()%><br>
  Boolean Cookies = <%=Request.Browser.Cookies.ToString()%><br>
  Boolean Crawler = <%=Request.Browser.Crawler.ToString()%><br>
```

```
  Boolean Frames = <%=Request.Browser.Frames.ToString()%><br>
  Boolean JavaApplets = <%=Request.Browser.JavaApplets.ToString()%><br>
  Boolean JavaScript = <%=Request.Browser.JavaScript.ToString()%><br>
  Int32 MajorVersion = <%=Request.Browser.MajorVersion.ToString()%><br>
  Double MinorVersion = <%=Request.Browser.MinorVersion.ToString()%><br>
  String Platform = <%=Request.Browser.Platform%><br>
  Boolean Tables = <%=Request.Browser.Tables.ToString()%><br>
  String Type = <%=Request.Browser.Type%><br>
  Boolean VBScript = <%=Request.Browser.VBScript.ToString()%><br>
  String Version = <%=Request.Browser.Version%><br>
  Boolean Win16 = <%=Request.Browser.Win16.ToString()%><br>
  Boolean Win32 = <%=Request.Browser.Win32.ToString()%><br>

</body>
</html>
```

بالإضافة إلى إمكانية الوصول بهذه الطريقة يستطيع المطور الوصول إلى إعدادات التهيئة باستخدام الصف System.ConfigurationSetting و استعادة البيانات من أي قسم في التهيئة.

يجب الملاحظة هنا أن الغرض الذي تتم إعادته باستخدام ConfigurationSettings يعتمد على معالج القسم المعين لهذا القسم .

المثال التالي يوضح كيفية الوصول إلى بيانات التهيئة للقسم <customConfig> .

يفترض مثالنا أن معالج القسم سوف يعيد غرض من النمط CustomConfigSettings مع الخاصة Enabled

```
Dim config As CustomConfigSettings = CType(ConfigurationSettings("customconfig"),
CustomConfigSettings)

If config.Enabled = True Then
   ' Do something here.
End If
```

استخدام إعدادات التطبيق

قد يكون أحد الاستخدامات الأكثر أهمية لملفات التهيئة هو حفظ الإعدادات المخصصة للتطبيقات،مثل الإعدادات الخاصة بسلسلة المحارف الخاصة بتعريف المزود و معلومات الاتصال بقاعدة بيانات ما ، مسارات الملفات ،مواقع ملفات XML ...إلخ.

تتضمن الأقسام المعرفة تلقائياً (المحددة ضمن الملف machine.config) القسم <appSetting> الذي يمكن استخدامه لتخزين الإعدادات على شكل ثنائيات (اسم/قيمة) .

يظهر المثال التالي القسم <appSettings> الذي يعرف اتصال بقاعدة البيانات لتطبيق.

```
<configuration>
  <appSettings>
    <add key="pubs" value="server=(local)\NetSDK;database=pubs;Trusted_Connection=yes" />
    <add key="northwind"
value="server=(local)\NetSDK;database=northwind;Trusted_Connection=yes" />
  </appSettings>
</configuration>
```

و فيما يلي النص البرمجي للصفحة التي سنقوم بالاستفادة فيه من القيمة المحفوظة في ملف إعدادات التهيئة.

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%@ Import Namespace="System.Configuration" %>

<html>

<script language="VB" runat="server">

    Sub Page_Load(Src As Object, E As EventArgs)

        Dim dsn As String = ConfigurationSettings.AppSettings("pubs")

        Dim MyConnection As SqlConnection
        Dim MyCommand As SqlDataAdapter

        MyConnection = New SqlConnection(dsn)
        MyCommand = New SqlDataAdapter("select * from Authors", MyConnection)

        Dim DS As New DataSet
        MyCommand.Fill(DS, "Authors")

        MyDataGrid.DataSource= New DataView(DS.Tables(0))
        MyDataGrid.DataBind()
    End Sub

</script>

<body>

  <h3><font face="Verdana">Retrieving Configuration Data</font></h3>

  <ASP:DataGrid id="MyDataGrid" runat="server"
    BackColor="#ccccff"
    BorderColor="black"
    ShowFooter="false"
    CellPadding=3
    CellSpacing="0"
    Font-Name="Verdana"
```

```
  Font-Size="8pt"
  HeaderStyle-BackColor="#aaaadd"
 />

</body>
</html>
```

كما نلاحظ في المثال تم الوصول إلى قيم إعدادات التهيئة باستخدام التعبير:

```
Dim dsn As String = ConfigurationSettings.AppSettings("pubs")
```

الفصل الثامن عشر – الجزء الأول

عنوان الموضوع:

الخبء في ASP.NET

الكلمات المفتاحية:

خبء ، انتهاء ، صلاحية ، خبء الخرج ، الخبء الجزئي ، خبء البيانات

ملخص:

يعد الخبء من الآليات الضرورية و المستعملة في الكثير من المجالات لرفع الأداء. استفادت ASP.NET من هذه التقنية و قدمت مجموعة من الطرق لاستخدامها.

سنحاول خلال هذه الجلسة التعرف على هذه الطرق و كيفية تطبيقها.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- مفهوم الخبء و طرقه؛
- خبء الخرج
- الخبء الجزئي
- خبء البيانات

المخطط:

الخبء في ASP.NET

مقدمة عن الخبء

تُستخدم تقنيات الخبء بصورة واسعة في مجال الحوسبة لرفع الأداء، وذلك بالمحافظة على البيانات التي يتم الوصول إليها بشكل متكرر، أوالبيانات ذات الكلفة العالية (التي تستهلك الكثير من الموارد للوصول إليها) ضمن الذاكرة.
و إعادة HTTPيستخدم الخبء في مجال تطبيقات الوب للحصول على الصفحات أو البيانات عبر طلبات استخدام هذه البيانات دون عناء إعادة إنشاءها أو الحصول عليها.

ثلاث أنواع من الخبء، والتي يمكن استخدامها في تطبيقات الوب:ASP.NETتوفر

- خبء الخرج : حيث يتم خبء الاستجابات الديناميكية التي يولدها طلب ما؛
- الخبء الجزئي : حيث يتم خبء جزء من الاستجابة التي يولدها طلب ما؛
- خبء البيانات : حيث يتم خبء أغراض محددة برمجياً .

يكون خبء الخرج مفيداً في الحالات التي يمكن في خبء كامل الصفحة. فعلى المواقع التي يكون عليها تردد الطلب عالي، تكون عملية خبء صفحة و لو لدقيقة سبباً في تحسين كبير في الأداء. حيث يمكن تخديم الكثير من الطلبات لهذه الصفحة مباشرة دون تنفيذ النص الذي أنشأها .

في بعض الأحيان لايكون خبء كامل الصفحة عملياً– بسبب حاجة الصفحة مثلاً للتخصيص بترويسة خاصة بكل طلب. في هذه الحالة يكون من المجدي التعرف في الصفحة على الأغراض أو البيانات التي تحتاج كلفة (معالجة) عالية لإنشائها والصالحة ليتم خبئها. عندها يمكننا إنشاؤها و خبئها لمدة محددة من الزمن. كما يمكن أن تُستخدم طريقة الخبئ الجزئي لخبئ مقاطع من خرج الصفحة.

زمن الخبء:
يعد تحديد زمن الخبء أيضاً أحد القرارات بالغة الأهمية .
يمكن أن يتم تحديث البيانات بفواصل زمنية منتظمة أو بتحديد صلاحية البيانات بوقت محدد.
في هذه الحالة يتم إعطاء العناصر سند انتهاء صلاحية يحدد إزالة العنصر من الخبء في حال انتهاء صلاحيته.

يقوم النص البرمجي الذي يحاول الوصول إلى معلومات الخبء بالتحقق من وجود المعلومة المطلوبة في الخبء. و في حال عدم توفرها يتم إعادة إنشائها حسب الحاجة.

يدعم الخبء في ASP.NET الملحقات من نوع ملف أو مفتاح خبء بشكل يمكّن المطور من إنشاء عناصر خبء تعتمد على ملف خارجي أو على عنصر خبء آخر حيث تستخدم هذه التقنية لإنهاء صلاحية عنصر اعتماداً على أي تغيير في موارد معينة (ملف أو عنصر آخر).

خبء خرج الصفحة

يعد خبء الخرج أحد التقنيات الفعالة في زيادة الانتاجية للعملية طلب/استجابة بإجراء خبء للمحتوى الذي يتم توليده من الصفحات الديناميكية.

يكون خيار خبء الخرج مُفعّل تلقائياً في Asp.NET و لكن عملياً لن يتم خبء أي استجابة لأي طلب ما لم اتخاذ إجراء واضح لجعل الاستجابة قابلة للخبء.

لجعل استجابة ما قابلة للخبء لا بد من امتلاكها لسند صلاحية و انتهاء إضافة إلى قدرتها إلى على الوصول إلى الخبء.

يمكن إتمام هذه العملية باستخدام إحدى طريقتين :

- OutPutCache API (طريقة منخفضة المستوى)
- الموجه OutPutCach@.(طريقة عالية المستوى)

عند تفعيل خبء الخرج يتم إنشاء قيد خبء خرج لدى أول طلب بالطريقة GET للصفحة.
بعد هذا يتم تخديم جميع الطلبات (بالطريقة GET أو HEAD) التالية على هذه الصفحة من خبء الخرج لحين انتهاء صلاحية معلومات الخبء.

يتبع خبء الخرج كما ذكرنا سندات الانتهاء و الصالحية الخاصة بالصفحة . فإذا كان صفحة ما في خبء الخرج و تم تحديد سند الانتهاء ب 60 دقيقة من بداية الخبء، سيتم إزالة الصفحة بعد انقضاء 60 دقيقة من خبء الخرج.

في حال تم استقبال طلب آخر بعد هذا التوقيت سيتم إعادة تنفيذ النص البرمجي على الصفحة و يمكن عندها خبء الصفحة من جديد. يطلق على هذا النمط من انتهاء الصلاحية اسم انتهاء الصلاحية المطلق .

حيث يظهر الزمن OutPutCache@يوضح المثال التالي طريقة بسيطة لخبء الخرج باستخدام الموجه الذي تم توليد الاستجابة فيه.
لتعيين مدة الخبء ل 60 ثانية إضافة إلى تحديد قيمة الواصفة outPutCache@نلاحظ استخدامنا للموجه بمعنى أن الصفحة لن تختلف بحسب الطريقة المستخدمة سواء كان "non" إلى القيمة VaryByParam GET أو POST.

```
<%@ OutputCache Duration="60" VaryByParam="none" %>
<html>
 <script language="VB" runat="server">

  Sub Page_Load(Src As Object, E As EventArgs)
      TimeMsg.Text = DateTime.Now.ToString("G")
  End Sub

 </script>

 <body>

  <h3><font face="Verdana">Using the Output Cache</font></h3>

  <p><i>Last generated on:</i> <asp:label id="TimeMsg" runat="server"/>

 </body>

</html>
```

خبء الخرج

أما بالنسبة للتطبيقات التي تريد سيطرة أكبر على ترويسات HTTP المتعلقة بالخبء فعليها استخدام الآلية التي يزودها الصف System.Web.HttpCachePolicy .

```
Response.Cache.SetExpires(DateTime.Now.AddSeconds(60))
Response.Cache.SetCacheability(HttpCacheability.Public)
```

لجعل السند المستخدم من النوع المنزلق تجري إعادة تأهيل زمن الانتهاء في كل مرة يتم فيها طلب الصفحة و ذلك باستخدام الخاصة SlidingExpiration

```
Response.Cache.SetSlidingExpiration(True)
```

عند استخدام سند الانتهاء المنزلق يقوم الطلب الموجه إلى المخدم المصدر دائماً بإعادة استجابة.
يفيد الانتهاء المنزلق في الحالات التي يوجد فيها خبء يلبي حاجة الزبون، فإذا لم تنتهي صلاحية المحتويات يمكن الاستغناء عن طلبها من المخدم المصدر.

كذلك تحافظ ASP.NET على الطرق القديمة التي كانت مستخدمة في ASP وهي:

```
Response.CacheControl = "Public"
Response.Expires = 60
```

مثال :

في المثال التالي نلاحظ استخدام عنصر التحكم Timemsg لإظهار الزمن الحالي .
سيظهر هذا النص البرمجي نفس القيمة لعنصر التحكم هذا فيما إذا كانت الصفحة ماتزال ضمن الخبء لأنه لن يتم إعادة تنفيذ الصفحة مرة أخرى.

```
<%@ OutputCache Duration="60" VaryByParam="state" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<html>
 <script language="VB" runat="server">

   Sub Page_Load(Src As Object, E As EventArgs)
      Dim MyConnection As SqlConnection
      Dim MyCommand As SqlDataAdapter
      Dim ds As DataSet
      Dim queryState As String
      Dim selectCmd As String

      queryState = Request.QueryString("state")
      If queryState = Nothing
         selectCmd = "select * from Authors"
      Else
         selectCmd = "select * from Authors where state = '" + queryState + "'"
      End If

      MyConnection = New
SqlConnection("server=(local)\NetSDK;database=pubs;Trusted_Connection=yes")
      MyCommand = New SqlDataAdapter(selectCmd, MyConnection)
```

```
    ds = New DataSet()
    MyCommand.Fill(ds, "Authors")

    MyDataGrid.DataSource=new DataView(ds.Tables(0))
    MyDataGrid.DataBind()

    ' capture the time of the current request
    ' subsequent requests while we're still cached
    ' will  show the original time
    TimeMsg.Text = DateTime.Now.ToString("G")
  End Sub

</script>

<body>
  <h3><font face="Verdana">Using the Output Cache</font></h3>

  <b>Authors by State:</b>

  <table cellspacing="0" cellpadding="3" rules="all" style="background-
color:#AAAADD;border-color:black;border-color:black;width:700px;border-collapse:collapse;">
    <tr>
      <td><a href="outputcache3.aspx?state=CA">CA</a></td>
      <td><a href="outputcache3.aspx?state=IN">IN</a></td>
      <td><a href="outputcache3.aspx?state=KS">KS</a></td>
      <td><a href="outputcache3.aspx?state=MD">MD</a></td>
      <td><a href="outputcache3.aspx?state=MI">MI</a></td>
      <td><a href="outputcache3.aspx?state=OR">OR</a></td>
      <td><a href="outputcache3.aspx?state=TN">TN</a></td>
      <td><a href="outputcache3.aspx?state=UT">UT</a></td>
   </tr>
  </table>

  <p>

  <ASP:DataGrid id="MyDataGrid" runat="server"
    Width="700"
    BackColor="#ccccff"
    BorderColor="black"
    ShowFooter="false"
    CellPadding=3
    CellSpacing="0"
    Font-Name="Verdana"
    Font-Size="8pt"
    HeaderStyle-BackColor="#aaaadd"
  />

  <p>

  <i>Last generated on:</i> <asp:label id="TimeMsg" runat="server"/>
```

```
  </body>
</html>
```

الخبء الجزئي

بالإضافة إلى خبء الخرج لصفحة كاملة تقدم ASP.NET طريقة بسيطة لخبء خرج أجزاء معينة من محتويات الصفحة . تسمى هذه العملية بالخبء الجزئي.
تتم العملية بتحديد الأجزاء المراد خبئها بعنصر تحكم خاص بالمستخدم و تفعيل الخبء عليه باستخدام الموجه OutputCache@ حيث يتم تحديد الزمن بالثواني لخبء هذه الأجزاء على المخدم .

فعلى سبيل المثال يوعز التعبير التالي إلى Asp.NET بخبء الخرج لمدة 120 ثانية وبنَسخ الخبء اعتماداً على قيم CategoryID و SelectedID

```
<%@ OutputCache Duration="120" VaryByParam="CategoryID;SelectedID"%>
```

نقصد هنا بنَسخ الخبء، إنشاء نسخة من الصفحة لكل قيمة لــ categoryID أو SelectedID لأن القيم في الخبء لن تكون صحيحة بالنسبة لجميع الطلبات. أي أن الخبء الذي يتم الوصول إليه في حالة المعملين التاليين مختلف:

```
http://localhost/mypage.aspx?categoryid=foo&selectedid=0
http://localhost/mypage.aspx?categoryid=foo&selectedid=1
```

بالإضافة إلى دعم الموجه outPutCache للواصفة VaryByParam، فإنه يدعم الواصفة VaryByControl.
يتم نَسخ الخبء في VaryByControl اعتماداً على اختلاف عناصر التحكم ضمن عنصر التحكم الخاص بالمستخدم، في حين يتم نسخ الخبء في حالة VaryByParam اعتماداً على الثنائيات (اسم/قيمة) التي يتم إرسالها باستخدام GET أو Post:

```
<%@ OutputCache Duration="120" VaryByParam="none" VaryByControl="Category" %>
```

ملاحظة: لا بد من استخدام الواصفة VaryByParam بصورة صريحة حتى في حال عدم الرغبة بتفعيل نَسخ الخبء بناء على قيم المعاملات.

الخبء الجزئي

إذا احتوى عنصر التحكم الخاص بالمستخدم عنصر تحكم قائمة اختيار منسدلة سيتعدد الخبء لعنصر التحكم الخاص بالمستخدم بناء على القيم المختارة من عنصر تحكم القائمة هذا.

يمكن توطين عناصر التحكم بصورة تكرارية ضمن الصفحة (أي تعريف عنصر تحكم ضمن عنصر تحكم آخر) كذلك يمكن توطين خبء الخرج لعنصر تحكم مستخدم ضمن خبء عنصر خرج عنصر تحكم آخر.

تقدم هذا الميزة نموذج مركب يمكن من خبء أجزاء مكونة من مجموعات خبء جزئية.

يوضح المثال التالي كيف يتم خبء قسمين من قائمة باستخدام عناصر التحكم الخاصة بالمستخدم.

```
<%@ Register TagPrefix="Acme" TagName="Menu" Src="Menu.ascx" %>

<html>
  <body>
    <table>
      <tr>
        <td>
          <Acme:Menu Category="LeftMenu" runat=server/>
        </td>
        <td>
          <h1>Hi, the time is now: <%=Now%> </h1>
        </td>
        <td>
          <Acme:Menu Category="RightMenu" runat=server/>
        </td>
      <tr>
    </table>
  </body>
</html>
```

خبء البيانات

- توفر ASP.NET محرك خبء يمكن استخدامه ضمن الصفحات لحفظ واستعادة أغراض معينة عبر طلبات HTTP.

- يكون الخبء في ASP.NET خاص بكل تطبيق على حدة و يتم تخزينه في الذاكرة.

- تكون مدة حياة ذاكرة الخبء متعلقة بمدة حياة التطبيق، فعند إعادة تشغيل التطبيق سيتم إعادة إنشاء ذاكرة الخبء من جديد.

- تقدم ASP.NET واجهة برمجية بشكل فهرس بسيط يسمح للمستخدم بوضع الأغراض واستعادتها من الخبء.

السيناريو الأبسط للقيام بهذه العملية هو التالي:

```
Cache("mykey") = myValue
```

و لاستعادة الغرض من الخبء يمكن بسهولة كتابة:

```
myValue = Cache("mykey")
If myValue <> Null Then
   DisplayData(myValue)
End If
```

أما بالنسبة للتطبيقات التي تتطلب وظائف أكثر تعقيداً فتوفر ASP.NET مجموعة من الآليات مثل الكنس ، الانتهاء، و توابع الملفات و المفاتيح.

خبء البيانات

مثال:

يظهر المثال التالي الطريقة المبسطة لاستخدام الخبء حيث يقوم بتنفيذ استعلام على قاعدة البيانات و يقوم بخبء النتيجة ، و التي يتم استخدامها خلال فترة حياة التطبيق.

عند تشغيل هذا المثال سنلاحظ الرسالة في أسفل الصفحة. عند أول طلب للصفحة ستظهر أن النتيجة قد تم الاستحصال عليها من مخدم البيانات و بعدها سيتم إظهار رسالة تدل بأن المصدر هو معلومات الخبء

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<html>
  <script language="VB" runat="server">
```

```
  Sub Page_Load(Src As Object, E As EventArgs)

     Dim Source As DataView

      ' try to retrieve item from cache
      ' if it's not there, add it

      Source = Cache("MyDataSet")

      If Source Is Nothing

          Dim MyConnection As SqlConnection
          Dim MyCommand As SqlDataAdapter

          MyConnection = New
SqlConnection("server=(local)\NetSDK;database=pubs;Trusted_Connection=yes")
          MyCommand = New SqlDataAdapter("select * from Authors", MyConnection)

          Dim ds As New DataSet
          myCommand.Fill(ds, "Authors")

          Source = New DataView(ds.Tables("Authors"))
          Cache("MyDataSet") = Source

          CacheMsg.Text = "Dataset created explicitly"
      Else
          cacheMsg.Text = "Dataset retrieved from cache"
      End If

      MyDataGrid.DataSource=Source
      MyDataGrid.DataBind()
  End Sub

 </script>

 <body>

  <form method="GET" runat="server">

    <h3><font face="Verdana">Caching Data</font></h3>

    <ASP:DataGrid id="MyDataGrid" runat="server"
      Width="700"
      BackColor="#ccccff"
      BorderColor="black"
      ShowFooter="false"
      CellPadding=3
      CellSpacing="0"
      Font-Name="Verdana"
      Font-Size="8pt"
      HeaderStyle-BackColor="#aaaad" />
```

```
    <p>

    <i><asp:label id="CacheMsg" runat="server"/></i>

   </form>
  </body>
</html>
```

مثال 2:

في هذا المثال سيتم إظهار عنصر يعتمد على ملف XML.

الفكرة مشابهة للمثال السابق و لكن مصدر المعلومات هنا هو ملف XML ، عند إضافة سجل جديد إلى الملف باستخدام النموذج ضمن الصفحة سيتم إعادة إنشاء العنصر في الخبء.

```
<%@ Import Namespace="System.IO" %>
<%@ Import Namespace="System.Data" %>

<html>

  <script language="VB" runat="server">

    Sub Page_Load(Src As Object, E As EventArgs)
        If Not IsPostBack
            LoadData()
        End If
    End Sub

    Sub NewAuthorBtn_Click(Src As Object, E As EventArgs)
        If Not Page.IsValid
            AuthorMsg.Text = "Some required fields are missing"
        Else
            Dim fs As FileStream
            Dim reader As StreamReader
            Dim ds As DataSet
            Dim newAuthor As DataRow
            Dim writer As TextWriter

            ' open the file and read the current authors
            ds = New DataSet
            fs = New FileStream(Server.MapPath("authors.xml"), FileMode.Open, FileAccess.Read,
FileShare.ReadWrite)
            reader = New StreamReader(fs)
            ds.ReadXml(reader)
            fs.Close()

            ' append a row
```

```
        Try
          newAuthor = ds.Tables(0).NewRow()
          newAuthor("au_id") = AuthorId.Text
          newAuthor("au_lname") = LastName.Text
          newAuthor("au_fname") = FirstName.Text
          newAuthor("phone") = Phone.Text
          newAuthor("address") = Address.Text
          newAuthor("city") = City.Text
          newAuthor("state") = AddressState.Text
          newAuthor("zip") = PostalCode.Text
          newAuthor("contract") = Contract.Checked
          ds.Tables(0).Rows.Add(newAuthor)
        Catch Exc As Exception
          CacheMsg.Text = "Failed to create author with id = (" & AuthorId.Text & ")<br>" &
"Author already exists."
          Return
        End Try

        ' rewrite the data file
        fs = New FileStream(Server.MapPath("authors.xml"), FileMode.Create,
FileAccess.ReadWrite, FileShare.ReadWrite)
        writer = New StreamWriter(fs)
        writer = TextWriter.Synchronized(writer)
        ds.WriteXml(writer)
        writer.Close()

        Cache.Remove("MyData")
        LoadData()
      End If
    End Sub

    Sub RefreshBtn_Click(Src As Object, E As EventArgs)
      LoadData()
    End Sub

    Sub LoadData
      Dim Source As DataView

      Source = Cache("MyData")
      If Source Is Nothing
        Dim ds As DataSet
        Dim fs As FileStream
        Dim reader As StreamReader

        ' read the data from the XML source
        ds = New DataSet()
        fs = New FileStream(Server.MapPath("authors.xml"), FileMode.Open,FileAccess.Read)
        reader = New StreamReader(fs)
        ds.ReadXml(reader)
        fs.Close()
```

```
            Source = New DataView(ds.Tables(0))

            ' cache it for future use
            Cache.Insert("MyData", Source, New CacheDependency(Server.MapPath("authors.xml")))

            ' we created the data explicitly, so advertise that fact
            CacheMsg.Text = "Dataset created explicitly"
        Else
            CacheMsg.Text = "Dataset retrieved from cache"
        End If

        MyDataGrid.DataSource = Source
        MyDataGrid.DataBind()
    End Sub

  </script>

  <body>

    <form runat="server">

      <h3><font face="Verdana">File Dependencies</font></h3>

      <ASP:DataGrid id="MyDataGrid" runat="server"
        Width="900"
        BackColor="#ccccff"
        BorderColor="black"
        ShowFooter="false"
        CellPadding=3
        CellSpacing="0"
        Font-Name="Verdana"
        Font-Size="8pt"
        HeaderStyle-BackColor="#aaaadd"
      />

      <hr>

      <h3><font face="Verdana">Add New Author</font></h3>

      <asp:Label ID="AuthorMsg" Text="Fill in the required fields below to add a new author"
ForeColor="red" Font-Name="Verdana" Font-Size="10" runat=server />

      <p>

      <table>
        <tr>
          <td>Author Id:</td>
          <td><ASP:TextBox id=AuthorId Text="111-11-1111" runat=server/></td>
          <td><ASP:RequiredFieldValidator ControlToValidate="AuthorId" Display="Static"
ErrorMessage="*" runat=server/></td>
        </tr>
```

```
        <tr>
          <td>Last Name:</td>
          <td><ASP:TextBox id=LastName Text="Doe" runat=server/></td>
          <td><ASP:RequiredFieldValidator ControlToValidate="LastName" Display="Static"
ErrorMessage="*" runat=server/></td>
        </tr>
        <tr>
          <td>First Name:</td>
          <td><ASP:TextBox id=FirstName Text="John" runat=server/></td>
          <td><ASP:RequiredFieldValidator ControlToValidate="FirstName" Display="Static"
ErrorMessage="*" runat=server/></td>
        </tr>
        <tr>
          <td>Phone:</td>
          <td><ASP:TextBox id=Phone Text="555 555-5050" runat=server/></td>
          <td><ASP:RequiredFieldValidator ControlToValidate="Phone" Display="Static"
ErrorMessage="*" runat=server/></td>
        </tr>
        <tr>
          <td>Address:</td>
          <td><ASP:TextBox id=Address Text="One Microsoft Way" runat=server/></td>
          <td><ASP:RequiredFieldValidator ControlToValidate="Address" ErrorMessage="*"
Display="Static" runat=server/></td>
        </tr>
        <tr>
          <td>City:</td>
          <td><ASP:TextBox id=City Text="Redmond" runat=server/></td>
          <td><ASP:RequiredFieldValidator ControlToValidate="City" ErrorMessage="*"
Display="Static" runat=server/></td>
        </tr>
        <tr>
          <td>State:</td>
          <td><ASP:TextBox id=AddressState Text="WA" runat=server/></td>
          <td><ASP:RequiredFieldValidator ControlToValidate="AddressState" ErrorMessage="*"
Display="Static" runat=server/></td>
        </tr>
        <tr>
          <td>Postal Code:</td>
          <td><ASP:TextBox id=PostalCode Text="98052" runat=server/></td>
          <td><ASP:RequiredFieldValidator ControlToValidate="PostalCode" ErrorMessage="*"
Display="Static" runat=server/></td>
        </tr>
        <tr>
          <td>Contract:</td>
          <td><ASP:CheckBox id=Contract Checked runat="server"/></td>
          <td></td>
        </tr>
      </table>

      <asp:button Text="Add New Author" OnClick="NewAuthorBtn_Click" runat=server/>
<asp:button Text="Refresh List" OnClick="RefreshBtn_Click" runat=server/>
```

```
    <p>

    <hr>

    <p>

    <i><asp:label id="CacheMsg" runat="server"/></i></p>

   </form>
  </body>
</html>
```

الجزء الثاني

عنوان الموضوع:

ورشة عمل

الكلمات المفتاحية:

فضاء العينة، حدث بسيط، حدث أكيد، حدث مستحيل، الأحداث المُستقلة، الاحتمال

ملخص:

الغرض من هذا الجزء من الجلسة استعراض حل بسيط متكامل لمتجر بقالة الكتروني، بالطبع سيتم التركيز على النواحي المتعلقة بكتابة النص البرمجي للتطبيق.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- كيفية دمج العناصر التعليمة التي تمت تغطيتها خلال الجلسات الماضية ضمن إطار عمل تطبيقي

ورشة عمل

يشمل هذا الجزء من الجلسة تطبيق متكامل حول إنشاء واجهة بيع إلكترونية لمحل بقالة باستخدام ASP.NET و VB.NE .

تشمل المرحلة الأولى إنشاء الصفوف المناسبة للعمل على بيانات المتجر الالكتروني. سنقوم باستخدام VB.NET لإنشاء هذه الصفوف.

نقوم أولاً باستيراد فضاءات الأسماء المطلوبة

```
Top of Form
Imports System
Imports System.Data
Imports System.Data.SqlClient
Imports System.Collections
```

ثم نعرف فضاء أسماء جديد باسم Market يتضمن الصفوف InventoryDB، OrderItem، OrderList

```
Namespace Market

  Public Class InventoryDB

    Public Function GetProducts(Byval categoryID as Integer) as DataTable

        Dim sqlConnection1 as SqlConnection
        sqlConnection1 = new
SqlConnection("server=(local)\NetSDK;database=grocertogo;Trusted_Connection=yes")

        Dim sqlAdapter1 as SqlDataAdapter
        sqlAdapter1 = new SqlDataAdapter("Select * from Products where categoryid=" &
CStr(categoryID), SqlConnection1)

        Dim products As New Dataset
        sqlAdapter1.Fill(products, "products")

        Getproducts = products.Tables(0)
    End Function


    public Function GetProduct(productID As Integer) As DataRow

        Dim SqlConnection1 As SqlConnection
        SqlConnection1 = new
SqlConnection("server=(local)\NetSDK;database=grocertogo;Trusted_Connection=yes")

        Dim SQLAdapter1 As SqlDataAdapter
        sqlAdapter1 = new SqlDataAdapter("Select * from Products where productID=" &
CStr(productID), sqlConnection1)

        Dim product As New DataSet
        sqlAdapter1.Fill(product, "product")

        GetProduct = product.Tables(0).Rows(0)
    End Function


    public Function GetProductCalories(productID As Integer) As DataTable

        Dim SqlConnection1 As SqlConnection
```

```
        SqlConnection1 = New
SqlConnection("server=(local)\NetSDK;database=grocertogo;Trusted_Connection=yes")

        Dim SQLAdapter1 As SqlDataAdapter
        sqlAdapter1 = new SqlDataAdapter("Select * from ProductDetails where productID=" &
CStr(productID), sqlConnection1)

        Dim details As new DataSet
        sqlAdapter1.Fill(details, "details")

        GetProductCalories = details.Tables(0)
    End Function
  End Class




  public class OrderItem

    public m_ProductID As Integer
    public m_Quantity As Integer
    public m_Name As String
    public m_Price As Double


    public Sub New(ProductID As Integer, Name As String, Price As Double, Quantity As Integer)
      MyBase.New
      me.m_ProductID = ProductID
      me.m_Quantity = Quantity
      me.m_Name = Name
      me.m_Price = Price
    End Sub


    public Readonly Property ProductID As Integer
      Get
        ProductID = m_ProductID
      End Get
    End Property


    public Property Quantity As Integer
      Get
        Quantity = m_Quantity
      End Get

      Set
        m_Quantity = Value
      End Set
    End Property
```

```
    public Readonly Property Name As String
      Get
        Name = m_Name
      End Get
    End Property

    public Readonly Property Price As Double
      Get
        Price = m_Price
      End Get
    End Property

    public Readonly Property Total As Double
      Get
        Total = m_Quantity * m_Price
      End Get
    End Property

  End Class


  public class OrderList

    private m_Orders As Hashtable = new Hashtable
    private m_TaxRate As Double = 0.08

    public Readonly Property SubTotal As Double

      Get

        If (m_Orders.Count = 0) then
          SubTotal = 0.0
        End If

        Dim x as OrderItem
        For Each x in m_Orders.Values
          SubTotal += x.Price * x.Quantity
        Next x

      End Get
    End Property


    public Property TaxRate As Double
      Get
        TaxRate = m_TaxRate
      End Get

      Set
        m_TaxRate = TaxRate
      End Set
```

```
    End Property


    public Readonly Property Tax As Double
      Get
        Tax = SubTotal * m_TaxRate
      End Get
    End Property

    public Readonly Property Total As Double
      Get
        Total = SubTotal * (1 + m_TaxRate)
      End Get
    End Property

    public Readonly Property Values As ICollection
      Get
        Values = m_Orders.Values
      End Get
    End Property


' This is the Default property
    public Default Readonly Property DefaultProp(name As String) As OrderItem
      Get
        DefaultProp = CType(m_Orders(name),OrderItem)
      End Get
    End Property


    public Sub Add(value As OrderItem)
      If Microsoft.VisualBasic.IsNothing(m_Orders(value.Name)) Then
        m_Orders.Add(value.Name, value)
      Else
        Dim oI as OrderItem
        oI = Ctype(m_Orders(value.Name), OrderItem)
        oI.Quantity = oI.Quantity + 1
      End If
    End Sub

    public Sub ClearCart()
      m_Orders.Clear()
    End Sub


  End Class

End Namespace
```

سنقوم في الملف global.asax باستخدام نص برمجي لإنشاء مثيل عن orderList ووضعه ضمن متحول جلسة ليتم استخدامه من قبل المستخدم في الجلسة كبطاقة شراء.

```
<%@ Import Namespace="Market" %>

<script language="VB" runat=server>

   public Sub Session_Start()

      If Microsoft.VisualBasic.IsNothing(Session("ShoppingCart")) Then
         Session("ShoppingCart") = new Market.OrderList()
      End If
   End Sub

</script>
```

ثم سنبدأ بكتابة نص aspx (النص يشكل ملف aspx وحيد و لكننا هنا نقوم بتفصيل كل جزء على حدة).

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="Market" %>

<html>
<head>
   <title>GrocerToGo</title>
   <link rel="stylesheet"href="grocerstyle.css">

   <style>
      div.details { background-color:ffffcc; padding-top:15; padding-bottom:20; }
      div.details table { width:375; }
      div.details table td { font-family:Verdana; font-size:8pt; }
   </style>

   <script language="VB" runat=server>

      public Sub Page_Load(sender As Object, e As EventArgs)

         if (Not IsPostBack) Then

            ProductListing.SelectedIndex = 0

            UpdateProducts()
            UpdateShoppingCart()
         End If
      End Sub
```

نلاحظ في المقطع السابق أننا قمنا بكتابة النص البرمجي لتحميل الصفحة و تأكدنا في كل مرة من استخدام الطرق UpdateProducts و UpdateShoppingCart للتأكد من تحديث قائمة المنتجات في الصفحة و

العناصر في سلة الشراء و المخزنة في غرض الجلسة.

ثم سنقوم بكتابة النص البرمجي الخاص بالأحداث المرتبطة بالتفاعل مع واجهة التطبيق كاختيار عنصر من قائمة التصنيفات المتوفرة أو الضغط على زر الإضافة لعنصر إلى سلة الشراء..إلخ

```
    public Sub CategoryList_Select(sender As Object, e As EventArgs)
      CurrentCategory.Text = CategoryList.Items(CategoryList.SelectedIndex).Text
      UpdateProducts()
    End Sub


    public Sub ProductListing_Select(sender As Object, e As EventArgs)
      UpdateProducts()
    End Sub


    public Sub AddBtn_Click(sender As Object, e As ImageClickEventArgs)
      Dim productID As Integer
      productID = Int32.Parse(ProductListing.DataKeys(ProductListing.SelectedIndex).ToString())

      Dim market As InventoryDB
      market = new InventoryDB()

      Dim product As DataRow
      product = market.GetProduct(productID)

      Dim shoppingCart As Market.OrderList
      shoppingCart = Ctype(Session("ShoppingCart"), Market.OrderList)

      shoppingCart.Add(new Market.OrderItem(productID, CStr(product("ProductName")),
Double.Parse(product("UnitPrice").ToString()), 1))

      UpdateShoppingCart()
    End Sub


    public Sub Recalculate_Click(sender As Object, e As ImageClickEventArgs)

      ' Obtain Shopping Cart From Session State

      Dim shoppingCart As Market.OrderList
      shoppingCart = Ctype(Session("ShoppingCart"), Market.OrderList)

      ' Iterate over items in shopping cart (update cart with current row qty textbox value

      Dim i As Integer
      Dim qty As HtmlInputText

      for i = 0 To ShoppingCartList.Items.Count - 1
```

```
      qty = Ctype(ShoppingCartList.Items(i).FindControl("Qty"), HtmlInputText)
      Try
        shoppingCart(CStr(ShoppingCartList.DataKeys(i))).Quantity = CInt(qty.Value)
      Catch exc As Exception
      End Try
    Next i

    UpdateShoppingCart()
  End Sub


  public Sub ClearCart_Click(sender As Object, e As ImageClickEventArgs)

    ' Obtain access to Shopping Cart From Session State

    Dim shoppingCart As Market.OrderList
    shoppingCart = Ctype(Session("ShoppingCart"), Market.OrderList)

    ' Clear Items From Shopping Cart and then Update UI

    shoppingCart.ClearCart()
    UpdateShoppingCart()
  End Sub
```

فيما يلي النص البرمجي للتوابع المعرفة مسبقاً و الخاصة كما ذكرنا بتحديث قائمة المنتجات:

```
  Sub UpdateProducts()

    Dim market As New InventoryDB

    ' Update Product Listing at Bottom of Page

    Dim categoryID As Integer
    categoryID = Int32.Parse(CategoryList.Items(CategoryList.SelectedIndex).Value)
    ProductListing.DataSource = market.GetProducts(categoryID).DefaultView
    ProductListing.DataBind()

    ' Update Product Information

    Dim productID As Integer
    productID = Int32.Parse(ProductListing.DataKeys(ProductListing.SelectedIndex).ToString())

    Dim product As DataRow
    product = market.GetProduct(productID)

    Name.Text = product("ProductName").ToString()
    SelectedProdPicture.Src = product("ImagePath").ToString()
    ServingSize.Text = product("ServingSize").ToString()
    Servings.Text = product("Servings").ToString()
```

```
      ' Update Product Calory Information

      DetailsListing.DataSource = market.GetProductCalories(productID).DefaultView
      DetailsListing.DataBind()
    End Sub

    Sub UpdateShoppingCart()

      ' Update Shopping Cart UI from Basket Stored in Session State

      Dim shoppingCart As Market.OrderList
      shoppingCart = Ctype(Session("ShoppingCart"), Market.OrderList)

      SubTotal.Text = System.String.Format("{0:C}", shoppingCart.SubTotal)
      Tax.Text = System.String.Format("{0:C}", shoppingCart.Tax)
      Total.Text = System.String.Format("{0:C}", shoppingCart.Total)

      ShoppingCartList.DataSource = shoppingCart.Values
      ShoppingCartList.DataBind()
    End Sub

  </script>
```

يظهر في هذا الجزء تصميم الصفحة مع عناصر التحكم من جهة المخدم المستخدمة مثل قوائم الاختيار و الصور.

```
</head>
<body topmargin="0" leftmargin="0" marginwidth="0" marginheight="0">

  <form runat="server">

    <table cellspacing=0 cellpadding=3 bgcolor="DC6035" border=0 width="100%">
      <tr>
        <td align="left"><img src="images/logo.gif"></td>
        <td align="right">
          <a><img border=0 src="images/home.gif"></a>
        </td>
      </tr>
      <tr>
        <td align="right" class="select" colspan="2">

          <b>Select Category:  </b>

          <select id="CategoryList" style="width:75" runat="server">
            <option selected value="1">Milk</option>
            <option value="2">Cereal</option>
            <option value="3">Soda</option>
          </select>
```

```
                <asp:button text="Select" OnClick="CategoryList_Select" runat=server/>

            </td>
         </tr>
      </table>

      <table border=0 width=100% cellspacing=0 cellpadding=15 bgcolor="ffffcc">
         <tr>

            <td valign=top bgcolor=ffffcc>

                <p>

                <h3>
                   <b>Product Category: <asp:label id="CurrentCategory"
runat=server>Milk</asp:label><b>
                </h3>

                <table width="100%" cellpadding=0 cellspacing=0 >
                   <tr style="padding-left:12" >
                      <td align="center" style="border-style:inset;" bgcolor="EDBE7B" width=140>
                         <img id="SelectedProdPicture" runat=server>
                      </td>
                      <td align="center" bgcolor="ffffcc" style="padding-right:0;">

                      <div class="details">

                         <table cellpadding=1 >
                           <tr>
                             <td colspan=3>
                               <b><font face="Verdana" size=3><asp:label id="Name" runat=server>
</asp:label></font></b>
                             </td>
                             <td align=right>
                               <asp:imagebutton ImageUrl="images/addcart.gif"
OnClick="AddBtn_Click" runat=server/>
                             </td>
                           </tr>
                           <tr>
                             <td colspan=4 >
                               Serving Size <asp:label id="ServingSize" runat=server>  </asp:label>
                             </td>
                           </tr>
                           <tr>
                             <td colspan=4>
                               Servings Per Container <asp:label id="Servings" runat=server>
</asp:label>
                             </td>
                           </tr>
                           <tr>
```

```
                <td height=5 colspan=4 bgcolor="000000"></td>
              </tr>
            </table>

            <asp:repeater id="DetailsListing"  runat="server">

              <ItemTemplate>
                <table cellpadding=0>
                  <tr>
                    <td colspan=3>
                      <b><%# DataBinder.Eval(Container.DataItem, "Name") %></b>
                      <%# DataBinder.Eval(Container.DataItem, "Grams") %>
                    </td>
                    <td align=right>
                      <b><%# DataBinder.Eval(Container.DataItem, "Percent") %>%</b>
                    </td>
                  </tr>
                </table>
              </ItemTemplate>

              <SeparatorTemplate>
                <table cellpadding=0 >
                  <tr>
                    <td colspan=4 height=1 bgcolor="000000"></td>
                  </tr>
                </table>
              </SeparatorTemplate>

              <FooterTemplate>
                <table cellpadding=0 >
                  <tr>
                    <td colspan=4 height=5 bgcolor="000000"></td>
                  </tr>
                </table>
              </FooterTemplate>

            </asp:repeater>

           </div>
          </td>
        </tr>
      </table>

      <p>

      <table>
        <tr>
          <td class="products">

            <asp:datalist id="ProductListing" DataKeyField="ProductID" showheader=false
showfooter=false OnSelectedIndexChanged="ProductListing_Select" repeatdirection="horizontal"
```

```
borderwidth=0 runat="server">

                    <ItemTemplate>
                      <table>
                        <tr>
                          <td width="150">

                            <asp:imagebutton borderwidth=6 bordercolor="#ffffcc"
commandname="Select" ImageUrl='<%# DataBinder.Eval(Container.DataItem, "ImagePath") %>'
runat=server/>

                            <p>

                            <%# DataBinder.Eval(Container.DataItem, "ProductName") %>
<br>
                            <%# DataBinder.Eval(Container.DataItem, "UnitPrice", "{0:C}")
%><br>
                          </td>
                        </tr>
                      </table>
                    </ItemTemplate>

                    <SelectedItemTemplate>
                      <table>
                        <tr>
                          <td width="150">

                            <asp:imagebutton borderwidth=6 bordercolor="red"
commandname="select" ImageUrl='<%# DataBinder.Eval(Container.DataItem, "ImagePath") %>'
runat=server/>

                            <p>

                            <%# DataBinder.Eval(Container.DataItem, "ProductName")
%><br>
                            <%# DataBinder.Eval(Container.DataItem, "UnitPrice", "{0:C}")
%><br>
                          </td>
                        </tr>
                      </table>
                    </SelectedItemTemplate>

                  </asp:datalist>

                </td>
              </tr>
            </table>

          </td>

          <td width="315" valign=top class="cart" bgcolor="#EDBE7B">
```

```
        <h3>Your Shopping Cart</h3>

            <asp:datalist id="ShoppingCartList" DataKeyField="Name" borderwidth=0
runat="server">

              <HeaderTemplate>

                 <table width="100%">
                   <tr>
                     <td width=35>
                       <b>Qty</b>
                     </td>
                     <td width=175>
                       <b>Product</b>
                     </td>
                     <td width=50>
                       <b>Price</b>
                     </td>
                     <td align="right" style="padding-right:10">
                       <b>Total</b>
                     </td>
                   </tr>

              </HeaderTemplate>

              <ItemTemplate>

                   <tr>
                     <td width=35>
                       <input type=text size=1 id="Qty" runat=server value='<%#
DataBinder.Eval(Container.DataItem, "Quantity") %>'>
                     </td>
                     <td width=175>
                        <%# DataBinder.Eval(Container.DataItem, "Name") %>
                     </td>
                     <td width=50>
                        <%# DataBinder.Eval(Container.DataItem, "Price", "{0:C}") %>
                     </td>
                     <td align=right style="padding-right:10">
                        <%# DataBinder.Eval(Container.DataItem, "Total", "{0:C}") %>
                     </td>
                   </tr>

              </ItemTemplate>

              <FooterTemplate>

                 </table>

              </FooterTemplate>
```

```
                    </asp:datalist>

                    <table border=0 width="100%">
                      <tr>
                        <td colspan=4><hr></td>
                      </tr>
                      <tr>
                        <td width=52></td>
                        <td width=225 colspan="2" align="left">
                          <b>Subtotal</b>
                        </td>
                        <td align="right" style="padding-right:10">
                          <asp:label id="SubTotal" runat=server/>
                        </td>
                      </tr>
                    <tr>
                      <td width=52></td>
                      <td width=225 colspan="2" align="left">
                        <b>Tax</b>
                      </td>
                      <td align="right" style="padding-right:10">
                        <asp:label id="Tax" runat=server/>
                      </td>
                    </tr>
                    <tr>
                      <td width=52></td>
                      <td width=225 colspan="2" align="left">
                        <b>Grand Total</b>
                      </td>
                      <td align="right" style="padding-right:10">
                        <b><asp:label id="Total" runat=server/></b>
                      </td>
                    </tr>
                   </table>

                   <p>

                   <div id="CheckoutPanel" runat="server">
                     <center>
                       <asp:imagebutton borderwidth=0 OnClick="Recalculate_Click"
ImageUrl='images\recalculate.gif' runat=server/>
                       <asp:imagebutton borderwidth=0 ImageUrl='images\checkout.gif'
runat=server/>
                       <asp:imagebutton borderwidth=0 OnClick="ClearCart_Click"
ImageUrl='images\clear_cart.gif' runat=server/>
                     </center>
                   </div>

               </td>
           </tr>
```



```
        </table>
    </form>

</body>
</html>
```